Modélisation d'environnements urbains virtuels

Julien Perret

N° d'ordre: 3508

THÈSE

Présentée devant

devant l'Université de Rennes 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention INFORMATIQUE

par

Julien PERRET

Équipe d'accueil : Siames - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :
Modélisation d'environnements urbains virtuels

soutenue le 14 décembre 2006 devant la commission d'examen

MM. :	Gérard	HÉGRON	Rapporteurs
	Marc	NEVEU	
MM. :	Gwenola	THOMAS	Examinateurs
	Jacques	ZOLLER	
MM. :	Kadi	BOUATOUCH	Directeur de thèse
MM. :	Stéphane	DONIKIAN	Co-directeur de thèse

Toutes les méthodologies ont leurs limites, et la seule "règle" qui survit, c'est : "tout est bon".

Feyerabend

Je tient tout d'abord à remercier Marc Neveu, Professeur à l'Université de Bourgogne, qui m'a fait l'honneur de présider ce jury ainsi que d'assumer la charge de rapporteur. Je remercie aussi Gérard Hégron, professeur à l'École nationale supérieure d'architecture de Nantes, d'avoir bien voulu accepter la tâche de rapporteur. Je veux de plus leur manifester ma très sincère reconnaissance quant au sérieux et à l'esprit critique dont ils ont fait preuve dans cet exercice.

Merci beaucoup à Jacques Zoller, Professeur des Écoles d'architecture à l'École d'architecture de Marseille Luminy, et à Gwenola Thomas, Maître de conférences à l'Université de Bordeaux 1, d'avoir bien voulu juger ce travail. Merci infiniment Gwenola pour tes conseils et ton intérêt pour mes travaux.

Je remercie enfin Kadi Bouatouch, Professeur à l'Université de Rennes 1, qui a dirigé ma thèse, et Stéphane Donikian, Chargé de Recherche au CNRS, qui l'a coencadrée. Merci à vous deux de m'avoir donné l'opportunité de faire cette thèse à vos côtés.

Je remercie toute l'équipe SIAMES (aujourd'hui Bunraku) de m'avoir accueilli et supporté. Merci notamment à Bruno, Marwan, Sébastien, Pascal, Olivier et Claudie, Yann et Fabien. Au sein de cette équipe, j'ai eu l'occasion de rencontrer des personnes que j'ai la chance de pouvoir considérer comme mes amis aujourd'hui : merci Fabrice pour ton soutien, ton calme, ta patience, tes conseil, ton excellent whisky et ta succulente cuisine. Merci Thierry pour ta disponibilité et ton rire contagieux. Merci Annick pour des millions de raisons, mais surtout pour ton énergie, ta passion et ta créativité qui resteront pour moi une source d'inspiration. Merci Gérald. Je ne sais pas comment exprimer simplement tout ce que je te dois. Finalement, je remercie Romain, pour son indéfectible soutien, que dis-je, sa foi en moi, qui est pour beaucoup dans la réussite de cette thèse : merci, merci, merci.

Je n'oublie pas ceux que j'ai réussi à garder près de moi. Je vous remercie infiniment, Niot, Marie-Cécile, Cédric, Yoann, Gurvan, Nicolas, Élisa, Olivier, Tony, Michèle, François, Cécile M, Sylvain, Lucie, Aassif, Fabien, Cécile V, Éric, Géraldine, Antoine, Yann, Gaël, Benjamin, Stéphane, Ugo, Philippe, Yannic, Eudes et Valérie B. Je tient tout particulièrement a remercier l'extraordinaire david Bonnaud au vocabulaire superlatif excessivement developpé pour m'avoir aidé à trouver la formule superlative qui conviens a chacunes des formidables personnes qui m'ont épaulées mais aussi aimé lors des 4 années précédentes. Merci aussi à Charles et Sumant (et à leurs familles) de m'avoir accueilli à Orlando. Merci Jaroslav, Hikaru, Zoran, Nathan d'avoir rendu mon séjour à Orlando aussi mémorable. Je remercie aussi ceux qui ont pris le train en route. Merci Tom, Agathe, Estelle, Glenn, Ronan, Carole, Max, Aymeric, Laurence, Rozenn, François, Karine, Anne, Jérôme, Laure, Stéphane et tous les autres.

J'ai une pensée pour ceux que j'ai perdu en route. En espérant que nos chemins

se recroisent.

J’ai gardé une petite place à part pour Guillaume et Sabine. Guillaume, si quelqu’un peut raconter et apprécier le déroulement de ma thèse, je ne vois que toi. Merci pour les pauses café à 2h du matin, pour les longues discussions, les encouragements et la complicité. Merci Sabine pour les séances photos le dimanche et les balades photo, j’espère qu’on pourra reprendre ça bientôt. Merci aussi, bien sûr pour ton amitié.

Je remercie bien évidemment ma famille : mes parents et mes frères. Merci tout simplement d’avoir été là et de me donner jour après jour autant d’amour.

En tout dernier, je tiens à remercier Valérie pour tout ce qu’elle m’apporte au quotidien d’attention, de soutien et d’amour.

Table des matières

Chapitre 1

Introduction

> ***La cité est un discours, et ce discours est véritablement un langage : la ville parle à ses habitants et nous parlons notre ville, la ville où nous nous trouvons, simplement en l'habitant, en la parcourant, en la regardant. Cependant, le problème est de faire surgir du stade purement métaphorique une expression comme «langage de la ville». Il est très facile métaphoriquement de parler du langage de la ville comme on parle du langage du cinéma ou du langage des fleurs. Le vrai saut scientifique sera réalisé lorsqu'on pourra parler du langage de la ville sans métaphore.***
>
> *Roland Barthes [Bar53]*

Comment modéliser la ville ? Quels sont ses éléments ? Comment améliorer les outils informatiques permettant cette modélisation ? Jusqu'où peut-on isoler le travail de modélisation de la ville, descriptif, pratique, d'un travail, théorique celui-là, sur la ville ? Voici quelques-unes des questions qui sont posées par les travaux présentés dans cette thèse. Les environnements urbains sont en effet d'une telle complexité, qu'un travail sur leur modélisation informatique nous pousse à questionner les méthodes même de modélisation ainsi que les outils qui servent à les définir, les implémenter, les utiliser. L'étude proposée dans ce document s'attache aux outils informatiques disponibles, à en définir les limites lorsqu'il s'agit de modéliser la ville, et à proposer des solutions ou des pistes de solutions pour les dépasser. Finalement, c'est sans doute la pensée de Barthes [Bar53] qui résume le mieux cette démarche : si la ville est un discours, et j'en suis convaincu, alors le problème de la modélisation est bien celui de définir le langage de la ville. Pour parler le langage de la ville, de nombreux obstacles épistémologiques se dressent sur notre chemin, et pour les franchir, de nouveaux outils théoriques et pratiques sont nécessaires.

> ***L'image du marchand et du prince est parfois utilisée pour suggérer les deux grandes forces à l'œuvre dans la création et la transformation des villes. Les formes urbaines sont effectivement le résultat de l'action conjuguée et interactive de processus socio-économiques et de politiques sur des espaces différenciés par leur site et leurs héritages urbanistiques et culturels. Mais une autre dynamique les sous-tend, celle de l'utopie, qui est à travers toute l'histoire, la quête de la forme urbaine idéale. En fait, leur compréhension suppose une approche globale complexe, diachronique et systémique.***
>
> *Rémy Allain [All04]*

Comme le suggère Rémy Allain [All04], la ville possède différentes échelles et existe dans plusieurs dimensions. Elle nécessite à ce titre une approche globale complexe. Diachronique aussi, car si la ville est un phénomène linguistique, elle ne peut s'étudier qu'à travers ses évolutions dans le temps, ses transformations. Systémique enfin, puisque si la ville est un système, elle ne peut être analysée que par une approche pluridisciplinaire, arborescente, et à la lumière de ses activités, des informations qui y transitent, et de leurs interactions, c'est à dire de ses propriétés auto-organisatrices. Une telle approche n'est-elle pas tout naturellement systémique ? [Roc06] Nous essayons ainsi d'observer ici une analyse systémique de la ville comme *système vivant complexe*, un système ouvert, en relation constante avec son environnement.

Afin d'introduire les différents concepts liés à la ville, nous allons tout d'abord questionner la représentation de la ville (section 1.1). Le problème de la définition des éléments du système urbain est ensuite abordé (section 1.2). Les modèles de l'évolution du système urbain seront traités en section 1.3. Finalement, une brève discussion ainsi qu'une présentation du plan de cette thèse terminent ce chapitre (section 1.4).

1.1 Le système urbain et sa représentation

> ***L'architecture [...] est comme une grande sculpture évidée, à l'intérieur de laquelle l'homme pénêtre, marche, et vit.***
>
> *Bruno Zevi [Zev59]*

Comme le rappelle Philippe Boudon reprenant les pensées de Gaston Bachelard et Henri Poincaré, la ville est un environnement à la fois perçu, vécu et conçu. «Il distingue l'espace tel qu'il est quotidiennement vécu et l'espace construit pour le comprendre, le second étant, pourrait-on dire en utilisant une expression d'Henri Poincaré, un «espace représentatif» du premier, l'espace concret» [Bou03].

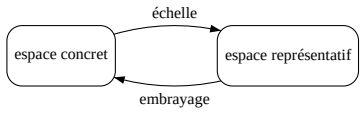


FIG. 1.1 – Illustration de la distinction entre échelle et embrayage.

Aussi, «si l'on retient le terme d'espace représentatif
défini par Poincaré, deux fonctions de représentation sont à distinguer : la fonction par laquelle l'espace est objet d'une représentation au sens mental, cognitif, du terme et une fonction par laquelle la représentation peut s'embrayer sur un réel et par là devenir réalité. Ces deux fonctions étant présentes dans l'usage du terme d'échelle, [Philippe Boudon a] cru nécessaire par la suite de les distinguer [...] en qualifiant la seconde d'*embrayage*, réservant celui d'*échelle* à la première» [Bou03]. La figure 1.1 illustre ces deux espaces et les fonctions de passage de l'un à l'autre. La distinction entre ces deux fonctions a ainsi pour but de clarifier et de pouvoir étudier indépendamment les processus de perception, liés à l'échelle, et ceux de conception, liés à l'embrayage.

Ainsi, pour Boudon, «[. . .] l'espace architectural est nécessairement le produit d'un travail précédant son existence et en conséquence [...]la connaissance de l'architecture ne saurait porter sur l'espace architectural comme objet premier, mais exige un *déplacement* de l'objet de l'architecturologie depuis l'espace architectural vers ce [que Boudon] a appelé *l'espace de conception*, espace dont l'architecturologie vise une modélisation sour forme d'un *espace architecturologique*.» [Bou03]

Il identifie ensuite les invariants de la conception architecturale : le nombre, l'esquisse, la perspective, le projet, la maquette.

> ***Au premier niveau, [...] le seul invariant est le nombre [...] : nombre de mètres carrés, nombre de pièces, etc. Au deuxième niveau, l'esquisse [...] correspondrait (à) cette géométrie des figures en caoutchouc qu'est la topologie, croquis de l'architecte qui traduit le besoin de ne définir que la structure de l'espace, connexions et continuités [...]. Au troisième niveau, l'architecte commencerait à se représenter l'espace perçu sous forme de perspectives [...]. Puis, au niveau suivant, le "projet" relèverait de la géométrie projective et utiliserait comme outil la géométrie descriptive [...]. Enfin, la maquette permettrait d'avoir en réduction un objet "semblable", au sens euclidien, à l'objet architectural projeté.***
>
> *Boudon, [Bou03]*

Si les travaux de Philippe Boudon se consacrent essentiellement à l'espace architectural, nous pensons qu'il n'est pas abusif d'étendre sa pensée à l'espace urbain, et ce malgré les réticences apparentes de celui-ci vis-à-vis de cette amalgame et notamment de la célèbre phrase de Le Corbusier : «Dès lors je confonds solidairement, en une seule notion, architecture et urbanisme.» Pour nous, en effet, ces deux espaces et leurs fonctions associées gardent tout leur sens à l'échelle (au sens commun du terme) de la ville, de la région, du pays.

Par ailleurs, nous pouvons nous demander s'il est possible d'étudier cet espace concret ainsi défini ou si nous sommes contraint de l'étudier indirectement à travers un espace représentatif propre à l'étude envisagée. En effet, il semble qu'il soit impossible de représenter le monde tel qu'il est, mais seulement ce que nos sens nous permettent de percevoir, notre entendement d'appréhender. Les théories portant sur la représentation de l'espace traitent donc de cet espace représentatif, et non de l'espace concret ; chacune d'elles est une vision, une analyse et une interprétation de l'espace en général et en particulier, ici, de l'espace urbain.

> ***Dans la mesure où les énoncés de géométrie parlent de la réalité, ils ne sont pas certains, et dans la mesure où ils sont certains, ils ne parlent pas de la réalité.***
>
> *Einstein, [Ein21]*

1.2 Les éléments du système urbain

> ***Ne jamais céder à la tentation de prendre au sérieux les problèmes concernant les mots et leurs significations. Ce qui doit être pris au sérieux, ce sont les questions qui concernent les faits : les théories et les hypothèses ; les problèmes qu'elles résolvent ; et les problèmes qu'elles soulèvent.***
>
> *Karl Popper, [Pop91]*

1.2.1 Le langage de l'architecture

L'étude des éléments du système urbain a commencé par l'étude des éléments de l'architecture. Vitruve, au premier siècle avant notre ère, est l'auteur du seul ouvrage de l'antiquité consacré à la théorie de l'architecture qui nous soit parvenu. Dans ses *Dix livres de l'Architecture* [VP95], il définit l'architecture comme une science qu'il constitue, en s'inspirant des grecs, pour plusieurs choses : le *Savoir*, l'*Ordonnance*, la *Disposition*, la *Proportion*, la *Bienséance* et la *Distribution*. Il s'appuie ensuite sur ces éléments pour établir une étude qui s'étend du choix de l'emplacement des édifices publics à la maçonnerie. Malgré la grande richesse de l'ouvrage, il faudra attendre 1544 pour que Guillaume Philandrier [Pal97] définisse le premier système formel en architecture. Il définit en effet l'ordre comme "succession d'éléments verticaux et horizontaux", posant ainsi les bases d'une vision plus normative de la conception architecturale et influença Vignole, Andrea Palladio, Léonard de Vinci (cf. 1.2) et bien d'autres. En particulier, Andrea Palladio est l'auteur du dernier et probablement du plus complet traité d'architecture de la Renaissance [Pal97]. On y trouve une véritable grammaire de l'architecture de la Renaissance, son vocabulaire et sa syntaxe. Ses travaux seront notamment le point de départ de nombreux travaux sur les grammaires génératives auquelles la section 3.4.1 est consacrée.

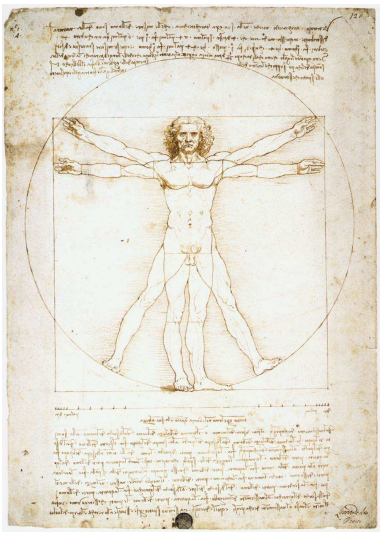


FIG. 1.2 – Dessin de Léonard de Vinci basé sur le livre second de Vitruve

Depuis Palladio, de nombreux architectes ont proposé leur définition de l'architecture en la partitionnant. Philippe Boudon [Bou03] remarque, à ce propos, la récurrence de la triade dans les nombreuses définitions de l'architecture. La figure 1.3 qui lui est empruntée en cite les plus connues. Néanmoins, ces découpages sont issus de doctrines et non de théories[1].

Alberti	voluptas	firmitas	commoditas
Blondel	agrément	solidité	commodité
SHA	venustas	firmitas	utilitas
Guimard	sentiment	logique	harmonie
Nervi	forme	structure	fonction

FIG. 1.3 – Classement des définitions de l'architecture selon Boudon. SHA : Société des Historiens de l'Architecture.

Sans pour autant prétendre définir l'architecture, les dictionnaires d'architecture proposent d'en définir les termes, le vocabulaire. Néanmoins, la définition d'un mot nécessite souvent de l'expliquer en utilisant d'autres mots du même lexique. Cette démarche n'est donc plus tout à fait lexicale, mais devient aussi syntaxique. On peut ainsi dire que le fameux ouvrage de Pérouse [dM00] propose une grammaire de l'architecture. Cette grammaire n'est néanmoins pas formelle et ne saurait être transformée directement et complètement en règles.

Afin de définir le langage de l'architecture, Bruno Zevi [Zev91] propose quant à

[1] Nous utilisons ici la terminologie proposée par Philippe Boudon : "La difficulté de cerner la question de la théorie [...] s'estompe déjà par la reconnaissance d'une distinction entre "théorie" en tant que support réflexif de la pratique et "théorie" en tant que visée de connaissance d'un objet, dans quelque champ que ce soit. Le terme de "doctrine" pour désigner la première, permettrait de réserver celui de "théorie" à la seconde interprétation."

lui une démarche résolument moderne visant à remettre en question les outils même de l'architecture : un langage alternatif de l'architecture mettant à jour les *invariants d'un code anticlassique*. Ainsi, ces invariants, remplaçant les invariants classiques (des grecs, de Vitruve, ... de l'académie), sont "sept invariants contre l'idolâtrie, les dogmes, les conventions, les phrases toutes faites, les lieux communs, les approximations humanistes et les phénomènes répressifs, sous quelque forme qu'ils se présentent, où qu'ils se cachent, qu'ils soient conscients ou inconscients". Zevi propose un langage radical, offrant une nouvelle vision de l'architecture, délivrant des habitudes de pensée, un code de l'architecture moderne au but avoué : "au lieu de parler sans fin *sur* l'architecture, finalement on parlera architecture". On retrouve ici la pensée de Barthes sur le langage de la ville appliquée à l'architecture. Mais même s'il est important, chez Zevi, d'intégrer l'édifice à la ville (c'est d'ailleurs le septième invariant), sa pensée se limite principalement à l'architecture, à ses propres codes, ses matériaux, ses formes.

Il en est tout autrement pour la démarche d'Alexander [AIS75, AIS77, Ale79]. Celui-ci propose en effet ce qu'il appelle un *langage de motifs* (Pattern Language) comme support à une théorie (*The Timeless Way of Building*) de l'architecture, de la construction et de l'urbanisme. Il s'agit en fait d'un langage composé d'archétypes ou motifs représentatifs des éléments de notre environnement. Ce langage est en réalité une méthode de conception basée sur l'idée que, pour qu'une ville (il en est de même pour un bâtiment) fonctionne, elle doit être conçue par ses habitants qui doivent, pour pouvoir communiquer, établir un langage commun. Nous voyons ici, que cette théorie est véritablement une doctrine puisque support de la pratique plutôt que visant à la connaissance. Néanmoins, cette doctrine met bien en valeur le problème, central, du langage de la ville et la nécessité de le définir. Ce qu'il propose, en réalité, c'est d'exprimer sous la forme d'un langage, l'utopie de la ville idéale propre à l'urbaniste, à l'architecte, au décideur, qui utilise cette méthode. On est bien ici dans le subjectif.

1.2.2 Le langage de la ville

> ***Un cadre physique vivant et intégré, capable de produire une image aigüe bien typée, joue aussi un rôle social. […] Un environnement ordonné de manière détaillée, précise et définitive peut interdire tout nouveau mode ou modèle d'activité.***
>
> *Kevin Lynch,* L'image de la cité *[Lyn60]*

En poussant plus loin encore cette subjectivité, Lynch [Lyn60] étudie l'image cognitive de la ville. Contrairement à Alexander qui utilise un langage de motifs, Lynch définit une image mentale, cognitive, formée par les cinq types d'éléments aujourd'hui incontournables de l'étude de la ville : les nœuds, les quartiers, les voies, les limites, et les points de repère. Il participe ainsi à l'effort initié, entre autres, par Jane Jacobs [Jac61] sur le problème du manque de *lisibilité* de la ville américaine. Il s'agit ainsi, au delà, effectivement, de rendre la ville lisible, d'affirmer que la ville existe aussi dans un espace qui lui est propre. Un espace de conception, comme Boudon le définit pour l'architecture, mais aussi un espace symbolique.

On peut ainsi voir des similitudes intéressantes entre l'image de la ville de Lynch et le discours de la ville de Barthes. Par ailleurs, le problème du mélange entre fonc-

tions techniques (un bâtiment doit *tenir debout* et supporter le poids de ses habitants) et fonctions symboliques (les décors de portes sculptées) a été traité depuis, notamment par Umberto Eco [Eco72] puis Philippe Fayeton [Fay99, Fay02]. Mais il apparaît aujourd'hui toujours aussi difficile, et l'une des démarches les plus réussies dans la séparation des fonctions est peut-être celle de Le Corbusier [Cor24, Cor30]. En effet, ce dernier a permis de séparer certaines fonctions constamment confondues en architecture : par exemple, les fonctions de *faire entrer la lumière* et de *donner un point de vue sur l'extérieur* attribuées à une fenêtre. Le Corbusier a ainsi proposé de faire des fenêtres hautes et larges, permettant la première fonction, et des fenêtres à hauteur d'yeux, focalisées vers les paysages offrant l'environnement.

Plus récemment, l'étude de la *syntaxe spatiale* s'incrit comme une méthode d'analyse des configurations spatiales urbaines. Initiée par Hillier [Hil96], cette étude propose d'étudier les propriétés et relations des espaces urbains et des bâtiments. Ce domaine de recherche, particulièrement riche, a donné lieu a des rapprochements avec la théorie de Lynch [DB03, Dal03, Pen03]. Romain Thomas [Tho05] propose d'ailleurs un modèle de cartes cognitives spatiales utilisant conjointement en partie ces deux théories.

Cette étude concernant les fonctions ne suffit néanmoins pas à une étude complète de la ville, et la morphologie urbaine propose une hiérarchisation de la ville.

1.2.3 Morphologie urbaine

> ***La morphologie urbaine est l'étude de la forme physique de la ville, de la constitution progressive de son tissu urbain et des rapports réciproques des éléments de ce tissu qui définissent des combinaisons particulières, des figures urbaines (rues, places et autres espaces publiques...).***
>
> *Rémy Allain [All04]*

La morphologie urbaine ne considère, par nature, que les formes de l'environnement urbain, ignorant ainsi les phénomènes sociaux en postulant "une certaine autonomie des formes et une logique intrinsèque de l'espace, qui rétroagit sur la société avec un décalage temporel." [Duc05] : la forme urbaine "est un terme qui ne recouvre pas entièrement le social et ne s'exclut pas de lui pour autant." [Ron02]

Les limites physiques (morphologiques) extérieures de la ville sont parfois floues [2], et pourtant géographiquement définies par des frontières : *la ville est un territoire*. De même, les bâtiments ont des frontières le plus souvent claires et répertoriées dans le cadastre. En effectuant cette démarche de façon systématique, il est ainsi possible de définir de façon hiérarchique les parties de la ville à travers ses éléments morphologiques. Rémy Allain en ressortira notamment les notions de *macroforme*, *plan*, *maillage*, *îlot*, *parcellaire* et *bâti* [All04]. On retrouve le problème de la décomposition de la ville de façon similaire chez Salingaros [Sal99, Sal00]. Ce dernier traite principalement des problèmes de cohérence des environnements urbains en étudiant les relations entre ses éléments en termes d'assemblage, de connections. Habraken [Hab00] va plus loin en étudiant les luttes de contrôle des différents acteurs de la construction et du maintien de la ville. Il définit alors une véritable hiérarchie de contrôle et de dominance.

[2] il suffit, pour s'en convaincre, de traverser les frontières entre Rennes et Cesson-Sévigné

Mais l'étude de la ville nécessite, au final, de se pencher sur les forces à l'œuvre dans son changement, et sur la nature de celui-ci.

1.3 L'évolution du système urbain

Le temps met tout en lumière.

Thalès

De nombreux facteurs influencent l'évolution du système urbain. Les réseaux sociaux [Fij02], par exemple, permettent d'étudier la structure sociale de la ville, son évolution et parfois son influence sur la construction urbaine. De façon plus globale, les cycles économiques et les facteurs socio-culturels déterminent les paramètres de cette construction. Mais, ce sont les utopies de la ville idéale qui influencent probablement le plus les processus de l'évolution urbaine [All04, PDD99]. Les utopies définissent les méthodes de la planification urbaine, les politiques de développement et les emplacements des villes. Malheureusement, ces utopies sont encore mal connues, et si nous comprenons mieux, aujourd'hui, les processus par lesquels nous prenons nos décisions, il n'en est pas toujours de même pour les processus appliqués au siècle dernier.

La ville est un environnement temporisé, et l'étude de ses cycles est primordiale à sa compréhension. De nombreux modèles d'étude existent aujourd'hui, allant de mesures statistiques des échelles de croissance [BPP02, Pum04] à l'étude de la fractalité urbaine [DFT04].

1.4 Discussion et plan de la thèse

La territorialité en tant que système de comportement [...] a évoluée de façon très semblable à celle des systèmes anatomiques. En fait, les différences de territorialité sont si bien reconnues aujourd'hui qu'elles servent à identifier les espèces, au même titre que les caractères anatomiques.

Edward T. Hall, La dimension cachée *[Hal71]*

Il existe un grand nombre de théories sur la ville, s'attachant tantôt à sa dimension humaine, sociale, symbolique, tantôt à sa dimension physique et morphologique. Les théories provenant de l'urbanisme, l'architecture, la géographie ou la sociologie sont extraordinairement riches, et nous ne prétendons pas en avoir fait un état de l'art représentatif. Nous avons néanmoins identifié certains des éléments constituant cet environnement complexe qu'est la ville. Parmi les éléments importants que nous avons voulu faire ressortir de ce panorama, nous souhaitons insister sur la ville définie comme discours. Comme nous l'avons vu, il est complexe de déterminer les éléments (ou mots) de ce langage, de même que leurs relations (ou structures syntaxiques), leurs fonctions (ou structures sémantiques), et leur évolutions. Nous nous limiterons ainsi dans cette étude au traitement (partiel) des questions relatives à la représentation des éléments du langage de la ville ainsi qu'à leur organisation. Pour cela, nous nous appuyons sur les différentes échelles hiérarchiques, la notion de réseau et d'opération de conception. Nous proposons ainsi une approche structurelle

de modélisation de la ville qui devra être complétée, dans de futurs travaux, par l'intégration, notamment, de la dimension symbolique afin d'aboutir à un modèle plus global et plus évolutif. Dans le chapitre suivant, nous étudions dans quelle mesure les approches informatiques de modélisation s'inspirent de ces théories et en prennent compte.

Dans la première partie de cette thèse (chapitre 1), nous avons dressé un panorama de différentes définitions de la ville ainsi que des différents angles d'étude potentiels de ses structures, échelles et éléments. Dans le chapitre suivant (chapitre 2), les différentes approches de modélisation de la ville et de ses éléments par des moyens informatiques sont présentées. Parmi ces méthodes, les méthodes dites *procédurales*, c'est à dire basées sur l'utilisation de fonctions pour la description d'objets, sont détaillées séparément au chapitre 3. Nos contributions sont ensuite présentées dans les chapitres 4 et 5. Le premier propose des contributions aux méthodes procédurales à partir de systèmes de réécriture. Le second présente un modèle et des algorithmes pour la représentation et la subdivision des environnements urbains virtuels. Le chapitre 6 conclut cette thèse et propose des perspectives pour de futurs travaux.

Chapitre 2

Approches de modélisation de la ville

Les applications nécessitant une représentation des environnements urbains sont nombreuses et de natures très variées. Ces applications recouvrent le projet d'architecte, l'aménagement urbain, la gestion du patrimoine, le cinéma et la télévision (pour les effets spéciaux et les films d'animation), les jeux vidéos. Chacune de ces application implique un domaine d'expertise différent, des éléments différents, des niveaux de détail différents. Chacune possède ses contraintes propres, des données propres et, le plus souvent, des outils propres.

Néanmoins, on pourra distinguer deux classes principales parmi ces applications. Ces deux classes nous amènent à considérer deux approches spécifiques de modélisation de la ville. La première consiste à modéliser une ville réelle, concrète, pour en obtenir un double virtuel aussi plausible que possible pour un domaine d'application donné. Dans cette première approche se situent le projet d'architecte, l'aménagement urbain, la gestion du patrimoine, et parfois le cinéma et les jeux vidéos. En suivant la deuxième approche, on modélisera des villes entièrement nouvelles, purement virtuelles. Cette deuxième approche concerne le plus souvent le cinéma, la télévision et les jeux vidéos.

La section 2.1 introduit la modélisation de villes réelles à travers les données et modèles existants. La section 2.2 traite des systèmes d'informations géographiques qui sont utilisés pour gérer les données urbaines. Les techniques de modélisation manuelles ou spécifiques sont présentées en section 2.3, suivies par l'utilisation de bases de connaissances pour la modélisation en section 2.4. Nous terminons enfin ce chapitre par une discussion sous forme de reformulation des problèmes de modélisation de la ville comme problème sémiotique (section 2.5).

2.1 Méthodes de modélisation basées sur des données

Pour créer des modèles virtuels d'environnements urbains réels, nous disposons de données plus ou moins objectives sur ces derniers. Les images satellites et les photographies aériennes sont, en l'occurrence, des sources d'information très riches. Elles nous informent, entre autres, de l'élévation du terrain, de l'hydrométrie, de la nature des sols. Par ailleurs, lorsqu'elles sont suffisamment précises, elles nous donnent des informations sur la disposition des rues, trottoirs, bâtiments, ainsi que de la végétation. Les photographies prises au sol permettent de reconstruire la géométrie des façades et de déterminer les textures des objets composant l'environnement urbain. Les techniques de relevé utilisant le laser sont, quant à elles, significativement plus précises, et permettent d'observer les détails des matériaux composant l'environnement construit.

Néanmoins, ces techniques, aussi puissantes qu'elles soient, ne nous donnent pas toujours des informations suffisantes pour reconstruire un environnement tridimensionnel plausible. En effet, les vues satellites et aériennes ne fournissent pas toujours des données suffisantes concernant la hauteur de bâtiments par exemple ; le laser nous fournit des nuages de points qu'il est parfois difficile à transformer en surfaces. Chacune de ces méthodes a ses avantages et inconvénients dont nous donnons ici un aperçu.

Faisons tout d'abord un rapide inventaire des types de données dont nous disposons. Du point de vue de la méthode de mesure, nous ne parlerons ici que de la photographie (section 2.1.1.1) et du laser (section 2.1.1.2). Ces deux technologies

FIG. 2.1 – Image satellite de Rome.

peuvent toutes deux être utilisées sous deux angles de prise de vue que nous étudions sous les noms de vues aériennes (section 2.1.2) et vues terrestres (section 2.1.3).

2.1.1 Les techniques de mesure

2.1.1.1 Photogrammétrie

La photogrammétrie est une technique qui permet de mesurer des objets en utilisant le principe de la *stéréovision* ou vision en relief. Utilisée par la vision humaine, la stéréovision nous permet de percevoir le relief des objets grâce aux différences entre les deux images des objets formées sur la rétine de nos yeux. De la même façon, la photogrammétrie est utilisée en vision par ordinateurs à l'aide de plusieurs photographies d'un même objet prises sous des points de vue différents. La difficulté majeure de la photogrammétrie consiste à identifier les points, droites et surfaces se trouvant dans plusieurs images à la fois. C'est le problème de la *mise en correspondance*. Ainsi, ces techniques requièrent souvent des images proches les unes des autres pour que la correspondance soit efficace. D'autres méthodes consistent à faire intervenir l'utilisateur aux phases clefs de la reconstruction.

Ceci étant dit, il existe de nombreuses alternatives pour acquérir les images qui permettront de reconstruire les objets. La plus simple consiste à disposer des appareils photo fixes à divers endroits (judicieusement choisis) autour de l'objet étudié, les images peuvent aussi être prises depuis un véhicule en mouvement à l'aide d'une caméra [ZS01, DTC00, MZ]. La photogrammétrie, par ailleurs, est souvent combinée à d'autres techniques comme le *GPS*[1] ou le laser pour situer précisément la position de laquelle l'image a été prise [FZ03].

[1] *Global Positionning System* ou Système de positionnement à l'échelle du globe.

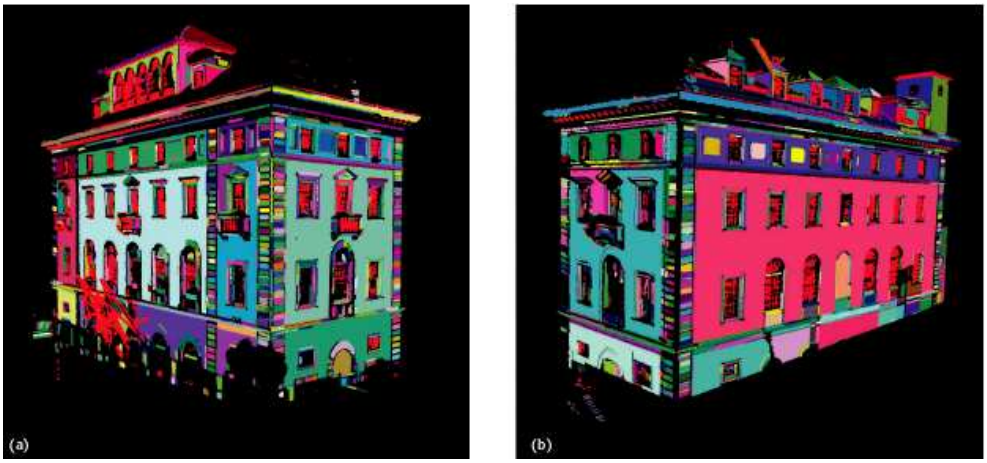


FIG. 2.3 – Segmentation de points 3D par classification. La Casa Italiana [Sin03]. Les surfaces connectées par l'algorithme sont de couleurs différentes pour plus de clarté.

La vision par ordinateur est un domaine très dynamique, et en faire un état de l'art n'est pas l'objet de ce document. Nous ne présenterons ainsi ici que des méthodologies simples et les résultats principaux appliqués aux environnements urbains.

2.1.1.2 Lidar

Le laser est devenu un outil important dans la reconstruction d'objets tridimensionnels. Le *Lidar*, pour LIght Detection And Ranging, désigne les systèmes utilisant la technologie laser pour la détection d'objets distants, ainsi que la mesure de leur position, vitesse et d'autres caractéristiques incluant les conditions atmosphériques (température, hydrométrie, vent, etc.). Cette technologie utilise le temps de réponse et la nature de l'onde lumineuse réfléchie par la surface des objets pour déterminer ces caractéristiques.

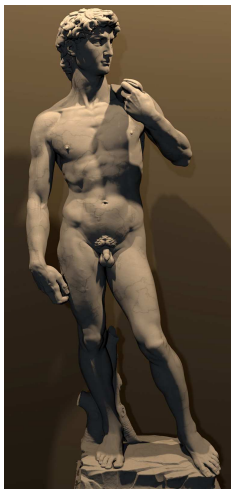


FIG. 2.2 – Le David de Michelange scanné au laser.

Le lidar est une technique d'une grande précision, néanmoins, les données récupérées sont difficiles à structurer. En effet, obtenir des segments et des surfaces à partir des nuages de points ne se fait pas sans s'exposer à des complications. Utilisée seule, cette technique est donc bien adaptée à la reconstruction d'objets présentant peu de surfaces planes comme c'est le cas, par exemple pour le projet *Michelangelo* [LPC$^+$00], visant à reconstruire des modèles de sculptures (voir figure 2.2). Par contre, les environnements urbains présentent de nombreuses surfaces planes pour lesquelles cette technique produit des détails qu'il faudra alors simplifier. C'est pourquoi le lidar est souvent utilisé de façon conjointe à la photogrammétrie. Néanmoins, de récentes recherches sont prometteuses quand à la segmentation des points 3D par classification des points présents sur une même surface [Sin03] comme illustré par la figure 2.3.

2.1.2 Vues aériennes

Au niveau géographique, l'acquisition des données sur la géométrie des objets urbains se fait principalement par la vectorisation d'images satellites redressées[2], de photos aériennes ou de fonds de cartes existantes. Cette vectorisation consiste à définir le contour des objets dans le plan (en 2D), soit manuellement (souvent à l'aide de relevés GPS), soit en utilisant la *photogrammétrie*. La figure 2.1 illustre une image satellite utilisée dans le contexte de *Google Earth*[3]. Cette application permet de visualiser un globe terrestre sur lequel sont appliquées des vues aériennes en temps réel. L'utilisateur peut ainsi se déplacer autour de la terre et se rapprocher d'un continent, d'un pays, d'une ville. Pour certaines villes, comme New-York ou Tokyo, des enveloppes simplifiées de bâtiments sont affichées. Ce type d'outil, utilisant des données urbaines (voir à ce sujet la section 2.2) variées telles que le nom des rues et des magasins ouvre de nouvelles perspectives en terme de cartographie et de marketing.

2.1.2.1 Extraction du réseau routier

L'extraction automatique du réseau routier peut être abordée grâce aux vues aériennes et à des connaissances sur le contexte des routes et la forme de celles-ci. Hinz [HBS+99] propose une méthode basée sur la reconnaissance du contexte des routes. En effet, dans les zones urbaines, les bâtiments et les véhicules, plus fréquents qu'en zone rurale, peuvent gêner la reconstruction du réseau. Ainsi, en détectant le contexte urbain ou rural, Hinz propose un algorithme de détection des bâtiments, des véhicules et des ombres portées permettant de rassembler efficacement les segments de routes détectés. Laptev [LML+00] propose quant à lui une approche utilisant l'extraction de routes saillantes pour extraire les parties occluses et les *snakes* pour l'extraction des axiales de ces routes. Ces derniers sont utilisés pour simuler le comportement d'une bande de largeur variable (le snake ou serpent) cherchant à se rapprocher au plus près des contours de l'objet d'intérêt (ici les routes) en utilisant un processus de minimisation de l'énergie.

Ces méthodes, si elles ne produisent pas des résultats parfaits, permettent de réduire considérablement les retouches manuelles à pratiquer sur le modèle.

2.1.2.2 Extraction de caricatures de bâtiments

De nombreuses méthodes basées sur la photogrammétrie, telle celle de Jibrini [JDPM], permettent de reconstruire des caricatures de bâtiments. La méthode de Jibrini consiste à retrouver un plan cadastral simplifié et utiliser la mise en correspondance pour évaluer la hauteur de chacun des bâtiment ainsi que la forme du toit (voir figure 2.4). D'autres méthodes, comme celle de Haala [HB99] permettent par ailleurs d'extraire les positions des arbres.

2.1.3 Vues terrestres

Les vues terrestres, prises du sol, sont des données particulièrement précieuses pour reconstruire des façades de bâtiment avec une grande fidélité ainsi que la vé-

[2] afin de corriger les déformations induites par les courbes de la surface de la terre ainsi que les défauts de l'angle entre l'objectif du satellite et la surface capturée

[3] earth.google.com

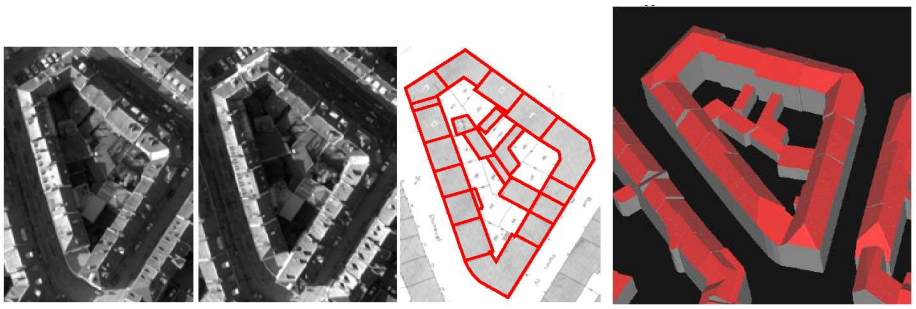


FIG. 2.4 – Extraction du cadastre (à gauche) et de caricatures de bâtiments (à droite).

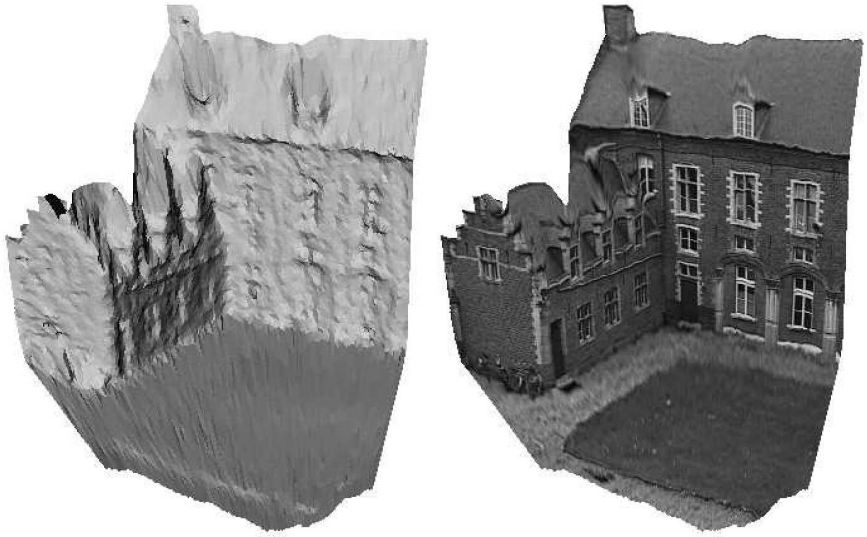


FIG. 2.5 – Modèle de la surface d'un bâtiment reconstruit, éclairé (à gauche) et texturé (à droite) [PKVG].

gétation. Néanmoins, elles ne contiennent pas d'information sur la toiture des bâtiments et présentent souvent des problèmes liés à l'occlusion (d'un arbre devant un bâtiment par exemple) [HYN03]. De plus, leur champ de vision limité incite à utiliser un grand nombre de vues. Par ailleurs, les modèles reconstruits sont souvent inutilement complexes. La figure 2.5, issue des travaux de Pollefeys [PKVG] montre un modèle reconstruit automatiquement. On remarquera le haut niveau de réalisme du modèle (visualisé sous certains angles), mais surtout la complexité de la géométrie d'un modèle pourtant (relativement) simple.

2.1.4 Les méthodes hybrides

Pour répondre à ce problème, Debevec[DTM96] propose une méthode, hybride, combinant images et géométrie dans le processus de modélisation. En effet, si les approches basées images utilisent des images (le plus souvent des photographies) pour déterminer automatiquement la structure d'un modèle 3D, les approches basées géométrie, quant à elles, placent la modélisation entre les seules mains de l'utilisateur. L'approche hybride de Debevec combine ces deux approches en deux étapes : l'une interactive, afin de construire un modèle de base simplifié de la scène, l'autre automatique, afin d'effectuer la correspondance stéréo entre images et modèle. La figure 2.6

FIG. 2.6 – Le fameux modèle du campanile [DTM96]. De gauche à droite : la photographie originale et les arêtes marquées, le modèle de base reconstruit, projection des arêtes du modèle sur la photographie et rendu synthétique obtenu.

illustre ce procédé de modélisation. À partir d'une ou plusieurs images, l'utilisateur marque des arêtes qui permettent de construire un modèle de base de la scène. Les photographies peuvent ensuite être appliquées sur les surfaces de ce modèle de base pour obtenir un rendu basé image. Reche [Rec05] a augmenté cette technique par des calques, édités par des infographistes, pour éviter les problèmes liés à l'occlusion.

Il est par ailleurs important ici de revenir sur un point que nous avons passé sous silence jusqu'ici. Nous avons fait l'hypothèse que l'objectif de ces méthodes était l'obtention d'un modèle géométrique. Ce n'est pas toujours le cas. En effet, dans le domaine de la modélisation et du rendu basé image, un modèle géométrique complet ou précis n'est pas toujours nécessaire.

2.1.5 Le rendu basé image

De nombreuses recherches portant sur l'utilisation d'images pour modéliser des environnements réels utilisent ces images, non seulement pour la modélisation, mais aussi pour le rendu[Rec05]. En effet, l'utilisation d'images réelles pour créer de nouveaux points de vue permet d'atteindre un haut niveau de réalisme. De plus, ces techniques peuvent être utilisées en temps-réel pour des environnements très complexes où le rendu basé géométrie est délicat. Certaines méthodes, comme les images panoramiques, se passent complètement de géométrie. Néanmoins, ces techniques ont deux inconvénients majeurs : les nouvelles vues pouvant être créées à partir des images d'entrée sont limitées aux objets entièrement capturés par celles-ci. De plus, le nombre d'images nécessaire à la modélisation d'un environnement urbain entier rend ces techniques inapplicables dans ce contexte, non seulement par le coût en temps et en calcul engendré par les prises de vues et leur traitement, mais aussi à cause de la taille considérable de stockage d'un tel nombre d'images. Pour finir, il faudrait disposer d'outils adaptés pour maintenir un tel système.

2.1.6 Conclusion

Comme nous l'avons vu, nous disposons aujourd'hui de méthodes puissantes pour reconstruire l'environnement urbain. Néanmoins, le problème le plus important qu'il reste à résoudre est le passage à l'échelle aussi bien géographique que temporelle. En effet, s'il est possible de reconstruire un modèle simplifié d'un ville de taille modeste comme Rennes, le passage à la région ou aux capitales inciterait aujourd'hui à baisser considérablement nos attentes. Par ailleurs, peu de travaux ont été menés concernant la cohérence de tels modèles dans le temps. La section suivante présente brièvement les systèmes d'informations géographiques qui servent à stocker et à manipuler les données urbaines.

2.2 Systèmes d'informations géographiques

2.2.1 Définitions

Un Système d'information géographique (SIG)

Il peut être défini comme un ensemble de matériels et logiciels autorisant le recueil, la saisie, la codification, la correction, la manipulation, l'analyse et l'édition graphique des données géographiques spatiales[DOCJ91].

Un Système d'Information Urbain (SIU)

C'est un ensemble de dispositifs cohérents de gestion de données urbaines associé à une représentation systématique de l'espace[eC89].

"Un SIU est une application particulière des SIG à la ville et l'urbain. Outre la différence d'échelle, tous les deux reposent sur les mêmes principes pour répondre à des préoccupations de l'ordre de *l'aménagement du territoire* dans le premier cas, de *l'urbanisation et la gestion des villes* dans le second."[Mah95]

L'information spatiale et urbaine peut être classifiée en deux types :

- les données graphiques (images ou vecteurs)
- les données non-graphiques (alphanumériques ou statistiques)

L'information géographique peut être classée en trois niveaux :

- le niveau géométrique (primitives graphiques)
- le niveau topologique (relation entre objets graphiques)
- le niveau sémantique (association entre la représentation graphique et le sens d'une information)

2.2.2 Constructions et Applications

Les Systèmes d'Information Géographique sont utilisés comme une représentation intermédiaire entre mesures et applications. Ces dernières, à l'origine principalement liées à la résolution de problèmes complexes d'aménagement et de gestion, recouvrent aujourd'hui le géo-marketing et les jeux-vidéos. A l'origine saisies à la main, les empreintes au sol des bâtiments, des rues, des parcelles, servent aujourd'hui à améliorer la reconstruction automatique d'environnements urbains. En contrepartie, les modèles ainsi créés enrichissent le système de géométrie et d'informations sémantiques. Un SIG est en fait un modèle spécifique mis en œuvre pour une zone géographique donnée.

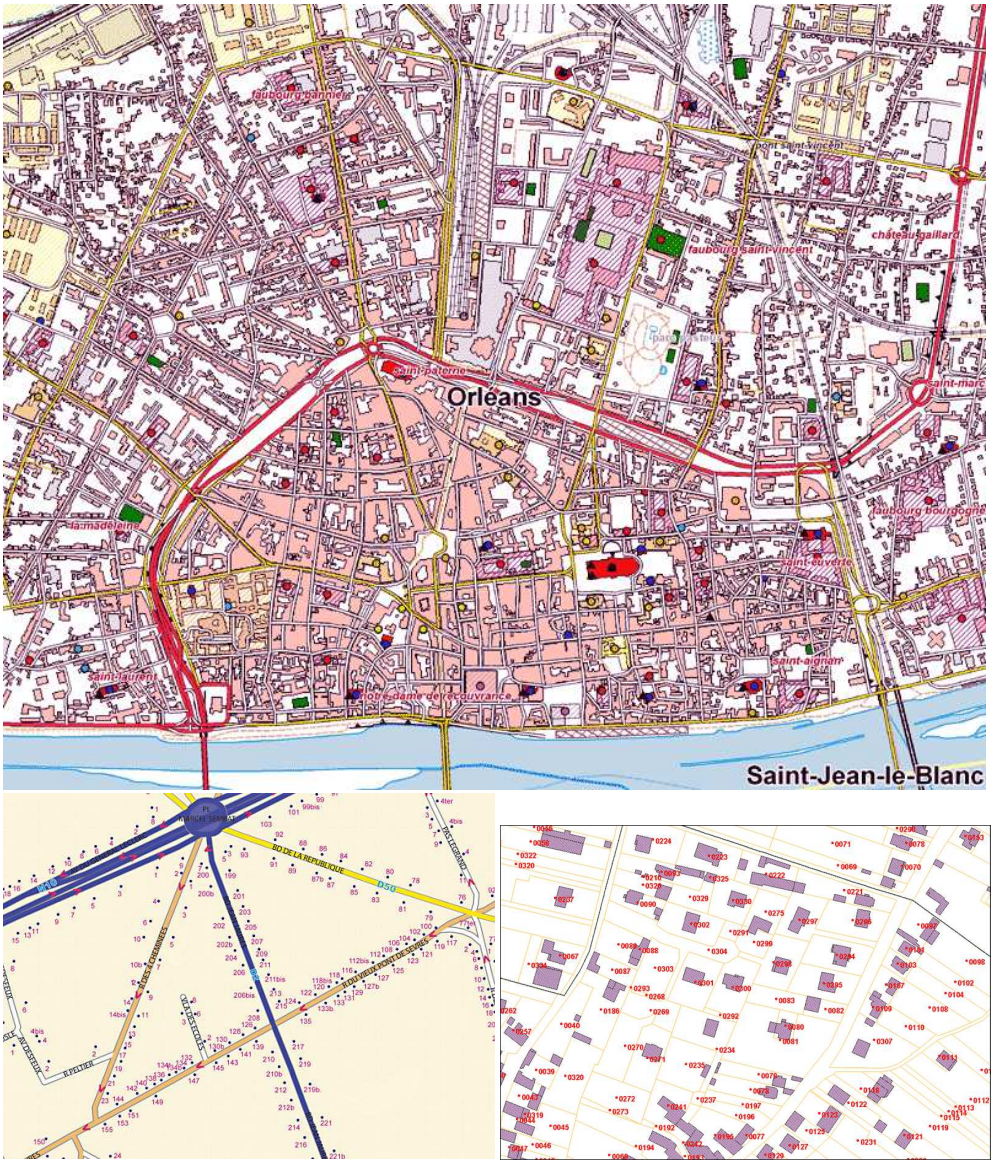


FIG. 2.7 – Trois bases de données faisant partie du SIG de l'IGN. En haut, la base de données topographique. En bas, de gauche à droite, les bases de données adresse et parcellaire.

2.3 Méthodes de modélisation géométrique spécifique

Les méthodes de modélisation spécifique désignent les méthodes pour lesquelles l'objet virtuel créé est spécifique, c'est à dire destiné à un propos précis, sans vocation de généralisation de ce modèle. Néanmoins, les outils de modélisation géométrique tendent à se spécialiser et à intégrer des connaissances propres à leur domaine de spécialisation. Dans cette section, nous présentons tout d'abord les méthodes manuelles, n'utilisant pas ou peu de connaissances, puis nous présentons les approches semi-automatiques, utilisant des informations spécifiques pour automatiser une partie de la modélisation. Les approches basées sur la représentation de connaissance spécialisée sont présentées dans la section suivante (2.4).

FIG. 2.8 – Modèle de ville modélisé manuellement.

2.3.1 Approches manuelles

Les approches dites manuelles désignent l'utilisation d'outils du commerce pour modéliser des environnements urbains. Ces outils, parmi lesquels Maya[4], 3DS Max[5], Blender[6] et AutoCAD[7], sont aujourd'hui suffisamment puissants pour permettre une telle démarche à l'utilisateur le plus courageux. Indépendamment des logiciels utilisés, ces méthodes varient principalement par leurs représentations des objets tridimensionnels. Ces représentations peuvent être regroupées en quatre familles :

- Solide : Les primitives sont des objets volumiques.
- Implicite : Les primitives sont les équations mathématiques de la surface des objets.
- Surfacique : Les objets sont des collections de surfaces.
- Basée points : Les objets sont composés de surfaces ponctuelles, c'est à dire de surfaces suffisamment petites pour pouvoir être approchées par des points.

L'approche du modèle virtuel de Tübingen [vVDBB98] utilise des plans d'architectes fournis par la ville, ainsi que des photographies prises sur le site pour reconstituer un modèle navigable en temps réel d'un bon niveau de réalisme (cf. figure 2.9). Pour ce faire, ils utilisent des outils du commerce tel Multigen[8] pour créer les modèles polygonaux simplifiés des bâtiments. Néanmoins, on ne peut pas encore ici parler d'une approche semi-automatique puisque chaque plan de chaque bâtiment doit être dessiné à la main (en utilisant les plans comme support), ainsi que les textures redressées et associées à la géométrie.

FIG. 2.9 – Vue du modèle virtuel de Tübingen [vVDBB98]. Le personnage et le panneau ont été rajoutés par les auteurs pour illustrer leur travail sur la perception des environnements urbains virtuels.

Ces approches peuvent aboutir à des résultats d'un grand réalisme (voir figure 2.8). Néanmoins,

[4] www.autodesk.fr/maya
[5] www.autodesk.fr/3dsmax
[6] www.blender.org
[7] www.autodesk.fr/autocad
[8] www.multigen.com

pour modéliser un environnement urbain dans toute sa complexité tout en gardant une base de donnée de dimension raisonnable et potentiellement modifiable, il est nécessaire de faire appel à des informations de plus haut niveau plutôt que les seules informations géométriques. En effet, les informations topologiques, hiérarchiques, méréologiques et sémantiques peuvent être particulièrement utiles pour manipuler les données urbaines [Don02].

2.3.2 Approches semi-automatiques

Les approches que nous appelons semi-automatiques, utilisent, à l'instar du modèle de Tübingen, des photographies. Néanmoins, ici, elles sont utilisées pour reconstruire l'environnement de manière semi-automatique, avec peu ou idéalement pas d'intervention humaine. Parmi celles-ci, nous présentons ici une approche basée sur des tuiles, les approches génératives, et finalement les approches dites *quasi-sémantiques* [Don02].

L'équipe de simulation urbaine [JLF96, JM99] (Urban Simulation Team ou UST) de l'université de Californie à Los Angeles (UCLA) ont conçu une approche de modélisation utilisant :

- des photographies aériennes numérisées,
- des photographies de façades redressées,
- des informations provenant d'un SIG.

FIG. 2.10 – Vue du centre ville de Los Angeles par l'équipe de simulation urbaine (UST) de l'université de Los Angeles (UCLA) [JLF96].

Les photographies aériennes, couplées aux informations géographiques permettent la reconstruction d'un modèle simplifié contenant routes et bâtiments, tandis que les photographies de façades sont plaquées sur la géométrie ainsi obtenue (cf. figure 2.10). L'algorithme utilisé pour la reconstruction des routes et des trottoirs est basé sur les tuiles illustrées par la figure 2.11. Une tuile est créée à chaque intersection de deux routes, ce qui permet la définition des zones d'intersection, des trottoirs et des blocs, contenant les bâtiments.

Malgré une automatisation plus grande que celle utilisée pour définir le modèle de Tübingen, cette méthode requiert toujours un travail manuel important. Néanmoins, nous voyons dans ces travaux une ébauche de structuration de la ville ainsi que l'utilisation de SIG. Par ailleurs, la partie la plus coûteuse de ces méthodes est la modélisation manuelle des bâtiments, notamment le redressage des textures de façade.

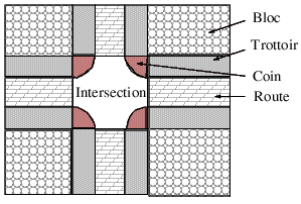


FIG. 2.11 – Tuile du modèle de l'UCLA.

2.3.2.1 Approches génératives

Pour certaines applications, il est possible de sacrifier la fidélité du modèle pour faciliter sa production. En effet, lorsque l'objectif de l'application n'est pas de reproduire avec précision un environnement réel, on peut se contenter de modéliser des prototypes de bâtiments. Ainsi, le nombre de photographies à récolter ainsi qu'à redresser est réduit, et, à partir d'une empreinte au sol,

les bâtiments sont facilement générés. Ces méthodes utilisent en fait des méthodes procédurales pour aider une modélisation qui, par ailleurs, reste manuelle.

Le laboratoire MAP-Aria propose, dans cette démarche, la reconstitution de la "Cité Industrielle" de Tony Garnier, architecte Lyonnais [MBST01]. La "Cité industrielle", est un projet de cité décrit à l'aide de plans, dessins et aquarelles, devant abriter 30000 habitants. Cette description inclue notamment toutes les habitations, structures administratives, industries, commerces,... mais aussi routes, fleuves, voies ferrées et l'environnement. La modélisation effectuée dans le cadre de ce travail, essentiellement manuelle, offre la possibilité de déambuler en temps réel dans un environnement qui n'a jamais existé que sur le papier. Il est par ailleurs intéressant de noter qu'en utilisant les plans dessinés par l'architecte, cette approche combine une modélisation déclarative proche des méthodes basées images (voir section 2.1) et une modélisation procédurale.

Un rendu non-photoréaliste multi-échelles : Afin de restituer au mieux les ambiances propres aux effets graphiques recherchés par l'architecte, un rendu non-photoréaliste est utilisé.

Les différents niveaux de représentation : En utilisant les aquarelles de Tony Garnier numérisées comme textures échantillonnées, Marsault et al. [MBST01] proposent une représentation hiérarchique de l'environnement. En particulier, les trois niveaux hiérarchiques apparaissent :

- *A l'échelle du territoire*, le bâti et la végétation sont approchés par des parallélépipèdes englobants habillés de textures sous-échantillonnées, les textures de terrain sont petites (128×128).
- *A l'échelle du quartier*, le bâti est représenté par son niveau de détail intermédiaire, les textures de terrain sont de 256×256.
- *A l'échelle de l'habitation*, les éléments géométriques caractéristiques apparaissent (trottoirs, mobilier urbain,...), les textures de terrain atteignent leur meilleure résolution : 512×512.

FIG. 2.12 – La "Cité Industrielle" de Tony Garnier [MBST01]. À gauche, une parcelle du plan général du site dessiné par Tony Garnier, et la même parcelle après texturage non photo-réaliste. Au milieu, une vue aérienne de la cité virtuelle. À droite, un exemple de niveaux de détail pour la salle des auditions.

La modélisation du terrain et des édifices a donc ici été automatiquement effectuée. Par contre, le placement des bâtiments et du réseau urbain reste manuel. En ce qui concerne ce dernier, MultiGen propose un ensemble d'outils permettant de modéliser la géométrie de la route à partir de profils horizontaux et verticaux ainsi que du rehaussement. D'autres recherches sont menées concernant la modélisation du réseau urbain à partir d'objets procéduraux. Entres autres, Bailey [BJPW99] définit les

Logical Road Network comme une encapsulation des routes de MultiGen y ajoutant des propriétés symboliques et des fonctions nouvelles (gestion des intersections, de la signalisation, …).

2.3.2.2 Approches quasi-sémantiques

Les approches de modélisation utilisant des vue aériennes et terrestres arrivent aujourd'hui, comme nous l'avons vu en section 2.1, à un haut degré de réalisme. Cependant, certaines applications requièrent des données de plus haut niveau sémantique et/ou des modèles plus simples pour la navigation temps-réel. L'application proposée par l'équipe SIAMES [Tho99] en collaboration avec la société IVT combine ces deux contraintes. En effet, l'objectif de cette application est la visualisation en temps-réel d'humanoïdes autonomes en environnement urbain informé. Une approche (quasi) sémantique est justifiée dans ce cadre puisque les piétons, afin d'être autonomes, nécessitent des information autres que la géométrie.

FIG. 2.13 – Vue du modèle virtuel de Rennes par la société IVT.

Ainsi, l'environnement de modélisation *VUEMS* [Don97, Tho99] est utilisé pour traiter la base de données (au format Ascodes) fournie par IVT (cf. figure 2.13) et permet l'introduction du réseau routier, de la signalisation, du mobilier urbain, des bâtiments, des places et des parcs [Tho99]. L'environnement ainsi construit est utilisé par la simulation de centaines de piétons autonomes et la navigation temps-réel. Nous ne nous intéresserons pas ici à la partie simulation du modèle, mais aux représentations de l'environnement urbain. Ainsi, afin de représenter la voirie, chaque rue est décomposée en tronçons. De plus, un corpus de 12 types de tronçons de réseau routier a été défini et est illustré par la figure 2.14. Chaque tronçon est ensuite lui-même composé de tronçons élémentaires illustrés par la figure 2.15 autour de la notion d'axiale. Cette dernière permet de construire le graphe de connexité entre les différents tronçons. Pour compléter le graphe ainsi constitué, un tissu urbain composé de bâtiments, de parcs et d'autres zones de circulation est ajouté, ainsi qu'un second graphe, de passage, qui est utilisé en complément des axiales pour la navigation des piétons virtuels [Don04].

2.4 Méthodes basées sur la représentation de connaissances

Nous avons, dans ce chapitre, traité des méthodes spécifiques, et, parmi les métamodèles, des méthodes procédurales et déclaratives. Néanmoins, comme nous l'avons vu, les méthodes procédurales et déclaratives n'étant pas diamétralement opposées mais plutôt complémentaires, elles peuvent être utilisées conjointement. En effet, les méthodes déclaratives tentent de décrire la nature de l'objet tandis que les méthodes procédurales traitent de la construction de celui-ci. Il suffit alors de considérer la construction de l'objet du point de vue de la connaissance que nous en avons pour trouver un terrain d'entente entre ces deux méthodes. Le modèle résultant correspond alors à la définition d'un système expert.

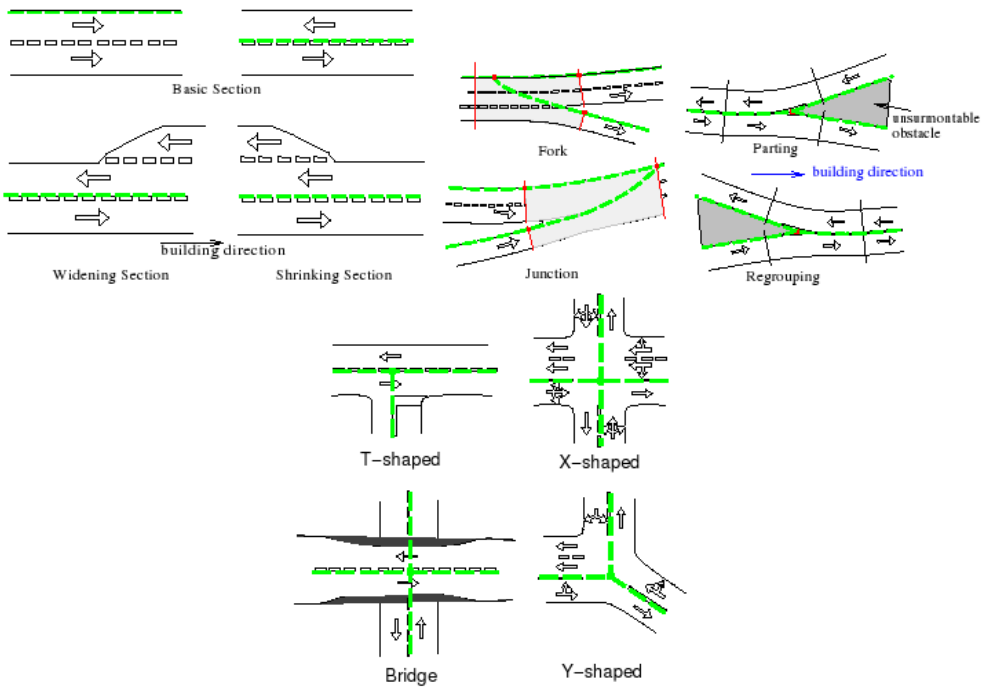


FIG. 2.14 – Différents types de tronçons d'un réseau routier dans VUEMS [Don04].

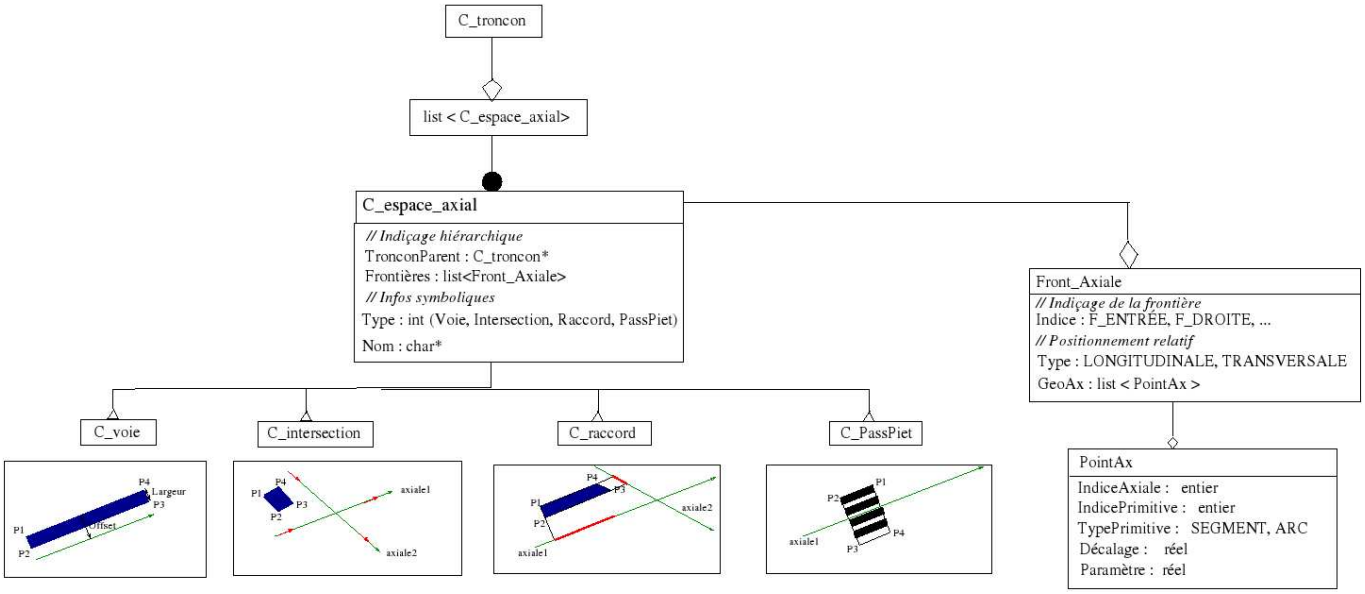


FIG. 2.15 – Différents types d'espaces axiaux dans VUEMS [Don04].

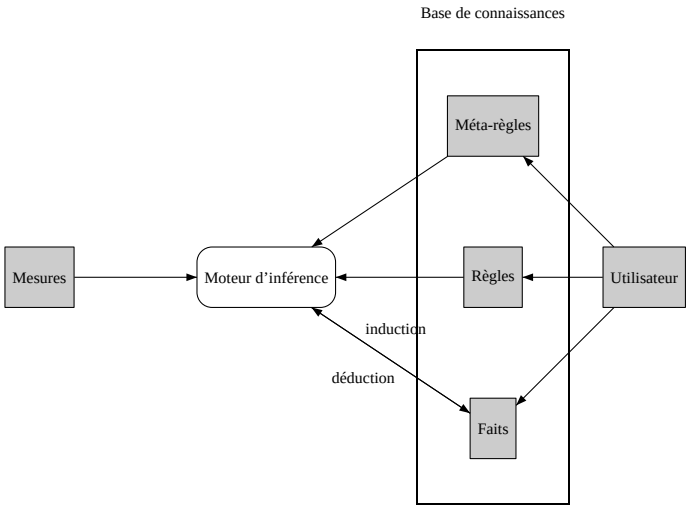


FIG. 2.16 – Architecture d'un système expert.

2.4.1 Définition d'un système expert

Le modèle que nous obtenons possède, de par sa nature procédurale, un ensemble de règles de production : *la base de règles*. Ces règles, faisant partie d'un modèle déclaratif, possèdent une sémantique liée aux informations qu'elles nous apportent. De plus, les connaissances déclaratives sur les objets s'expriment grâce à *une base de faits*. *La base de connaissances* regroupe la base de règles et la base de faits. Le système expert est un outil dont l'objectif est de reproduire une partie des mécanismes cognitifs d'un expert dans le domaine étudié. Ainsi, le système expert contient, en outre de la base de connaissances, un *moteur d'inférence* qui traite, à proprement parler, les connaissances du système. De plus, la *base des méta-règles* permet d'exprimer de la connaissance sur les règles et permet ainsi de contrôler le fonctionnement du moteur d'inférence. La figure 2.16 illustre l'architecture générale d'un système expert.

Il existe deux types de fonctionnement principaux du moteur d'inférence : le *chaînage avant* et le *chaînage arrière*. Le chaînage avant utilise une règle pour déduire de nouveaux faits, tandis que le chaînage arrière part des faits pour en induire les causes potentielles. Le chaînage avant constitue la partie procédurale du système, puisqu'il s'agit bien d'appliquer des règles. Le chaînage arrière constitue la partie déclarative du système : on assume que les faits sont vrais pour en déterminer la description du reste du système.

2.4.2 Un système expert pour l'architecture

Le laboratoire GAMSAU travaille depuis des années sur la connaissance architecturale et son utilisation conjointe à la mesure. Dans ce contexte, le projet PAROS [FBD97, Bla95, FBD00] propose un outil pour l'utilisation de mesures photogrammétriques guidées par une modélisation de la connaissance architecturale sous la forme d'une hiérarchie d'objets architecturaux et géométriques.

Les éléments du corpus architectural étudié, traitant des productions de l'antiquité grecque et romaine, sont regroupés dans une hiérarchie de classes d'objets illustrée

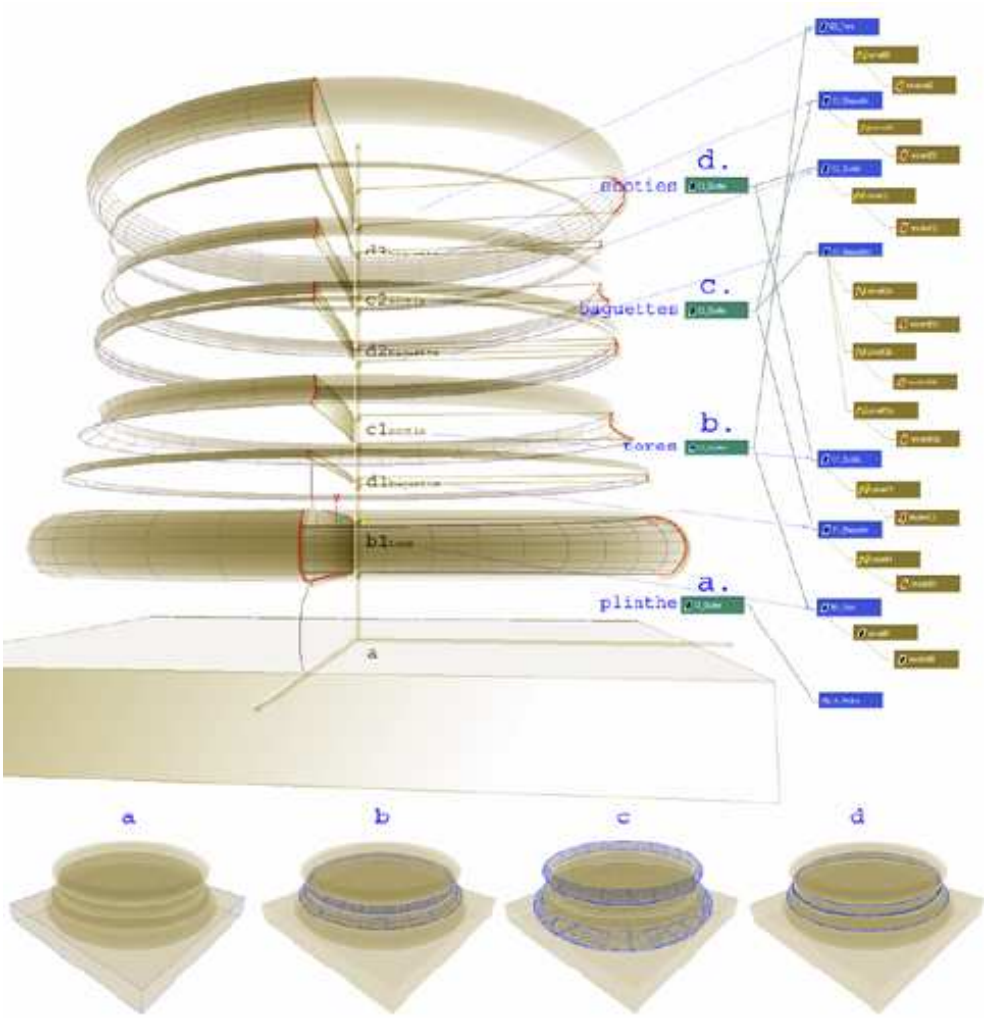


FIG. 2.17 – Modèle paramétré d'une base d'ordre corinthien [DGdL+03].

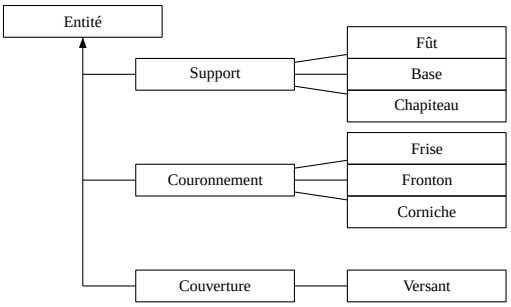


FIG. 2.18 – Extrait de la hiérarchie des classes d'objets pour le projet PAROS [Bla95].

FIG. 2.19 – Modèle reconstruit du Palais Royal à Paris [DGdL+03].

par la figure 2.18. L'édifice est ainsi un assemblage de tels objets regroupés sous la forme d'une hiérarchie de réseaux représentant leurs relations de composition. Chaque objet de cette hiérarchie est lié à une primitive géométrique. C'est la réduction de l'erreur entre le modèle théorique ainsi modélisé et la mesure qui permettra de valider ou d'invalider les hypothèses de la modélisation.

Dans la continuité de ces travaux, Dekeyser et al. [DGdL+03] proposent un système fonctionnant en deux phases. Dans une première phase, en ce qui concerne la mesure, ils utilisent un scanner laser couplé à une caméra, ce qui permet l'utilisation des techniques de reconstruction basées images et basées points. En effet, les images permettent de déterminer, avec l'aide de l'utilisateur, les motifs présents dans l'objet étudié et ainsi de structurer le nuage de points 3D. Dans une seconde phase, une base de connaissance constituée d'objets basés sur des courbes paramétrées (le vocabulaire, voir figure 2.17) et d'un ensemble de contraintes (la grammaire) sous la forme d'un script MEL (langage de script du logiciel Maya). Un tel système permet ainsi une modélisation rapide et précise de monuments architecturaux. La figure 2.19 illustre la reconstruction du *Palais Royal* à Paris.

2.4.3 Le passage à l'échelle urbaine

Ben Mahbous [Mah95], quant à lui, propose la construction d'un système d'information urbain (SIU) utilisant d'une part une représentation de la connaissance à base d'objets et des règles de production (de façon similaire aux travaux présentés ci-dessus), et d'autre part une base de données urbaines contextuelles.

L'utilisation de l'*approche Orientée Objet* permet d'une part d'intégrer des données géométriques, topologiques et sémantiques au sein d'une seule et même entité (un objet), et d'autre part de bénéficier des avantages de la programmation Orientée Objet (abstraction, héritage, factorisation, réutilisabilité, extensibilité, polymorphisme...). Un corpus correspond ici à une information sur un domaine applicatif donné (architecture, urbanisme, géométrie,...). La connaissance est alors représentée par un ensemble de corpus (d'objets) dits contextuels. Les comportements des objets d'un corpus sont modélisés par les règles de production associées.

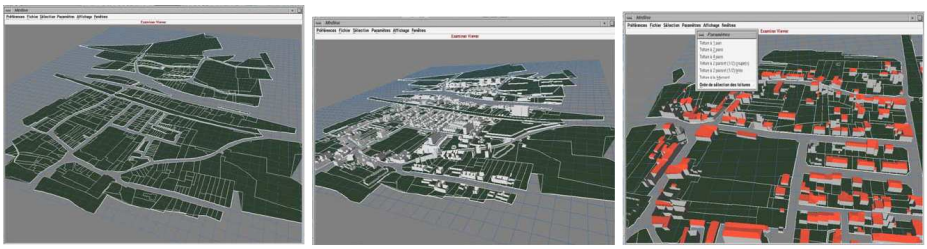


FIG. 2.20 – Reconstruction de volumétrie urbaines dans le cadre du projet Medina [PABCV00].

L'application présentée par Ben Mahbous est l'association de modules du projet *Remus*[Bou95, MZ94, MZ96], générateur de données morphologiques urbaines, et de l'application *Ricardo*, logiciel de cartographie thématique.

Le projet Medina [PABCV00] fait suite à ces travaux et présente une application de reconstruction de volumétrie urbaine et de toitures à partir de données cadastrales et de règles d'urbanisme (voir figure 2.20).

2.4.4 Discussion sur la représentation de données urbaines

De nombreuses autres recherches sur le thème de la représentation de connaissances liées à l'architecture et à l'urbanisme existent. Ainsi, Faucher [Fau01] propose une modélisation des enveloppes urbaines réglementaires grâce à une modélisation des règlements d'urbanisme produisant des contraintes sur la construction. Par ailleurs, Donikian [Don92] propose une approche de la conception architecturale basée sur la méréo-topologie. Liège [Liè97], pour finir, présente un système de modélisation pour la conception urbaine basé sur la résolution incrémentale de contraintes.

Mitchell [Mit90] présente un état de l'art de la modélisation de la connaissance architecturale. Ces méthodes posent en réalité la question des limites des possibilités de modélisation de la connaissance. Par ailleurs, de par la nature des solutions proposées, les approches que nous avons présentées utilisent des bases de connaissance supposées fixes au cours du temps. Néanmoins, notre compréhension du monde évolue rapidement et ces systèmes sont voués à devenir obsolètes s'ils ne sont pas façonnés pour être évolutifs. De plus, ces systèmes oublient le plus souvent l'évolution même des objets modélisés. Autran [Aut01] propose à ce sujet la prise en compte d'information spatio-historiques. Néanmoins, ces informations ne permettent pas de modéliser des phénomènes complexes tels que la rénovation urbaine, l'étalement urbain ou la densification. En effet, pour modéliser de tels phénomènes, il faut modéliser les processus sous-jacents qui régissent l'évolution de la ville. Les méthodes procédurales, que nous présentons au chapitre suivant, sont les plus adaptées pour cette tâche. Il est toutefois à noter que peu de travaux existent sur la représentation de tels processus, probablement à cause d'une mauvaise compréhension encore aujourd'hui de leurs natures, ainsi que de leur incroyable complexité. Quelques méthodes sont tout de même présentées en section 3.3.9. Nous souhaitons tout de même terminer ce chapitre par une reformulation qui permet, à notre avis, de mieux comprendre les différentes approches.

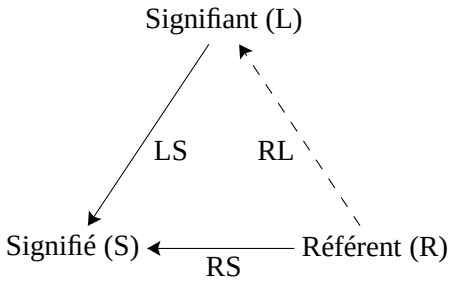


FIG. 2.21 – Le triangle sémiotique d'Ogden et Richards.

2.5 Une reformulation en termes sémiotiques

Il y a une sémiotique de l'architecture, de l'urbanisme et du design. L'architecture n'a certes pas pour fonction première de signifier : cet art entend d'abord protéger l'être humain, et lui offrir la possibilité d'exercer l'ensemble des activités qu'il souhaite mener. Il n'empêche que l'architecture et l'urbanisme communiquent aussi : par exemple une conception des rapports spatiaux et des rapports sociaux.

Klinkenberg [Kli96]

Le problème de la modélisation d'environnements virtuels, en particulier ici urbains, peut être reformulé comme un problème de cognition. En effet, le modèle virtuel d'un objet réel en est une représentation créée selon un modèle particulier. Nous ne nous attarderons pas ici sur la nature ou la validité de cette représentation mais sur le processus de construction de celle-ci. Pour ceci, il nous faut décomposer en étapes ce processus de modélisation. À cette fin, nous généralisons ce dernier en un processus cognitif.

2.5.1 Le triangle sémiotique

Rappelons tout d'abord la terminologie de Ogden et Richards, qui proposent le *triangle sémiotique* [OR23] comme modèle du processus de signification (voir figure 2.21). Les trois arêtes du triangle sont :

- Le référent, objet du processus cognitif.
- Le Signifiant, qui évoque le référent via des processus mentaux.
- Le Signifié, résultat du processus mental pour le sujet qui perçoit l'objet.

Appliqué au problème de la production de sens par le langage, le référent correspond à l'objet dont on parle, le signifiant au mot désignant l'objet, le signifié à l'image mentale que le sujet a de l'objet. Sur la figure 2.21, les flèches représentent les trois situations induites par le langage :

- *RS :* Le sujet *S* voit le référent *R*, ce qui déclenche le processus de production du signifié. Le sujet perçoit une chaise par exemple.
- *RL+LS :* Un locuteur *L*, qui perçoit le référent *R*, transmet l'information au

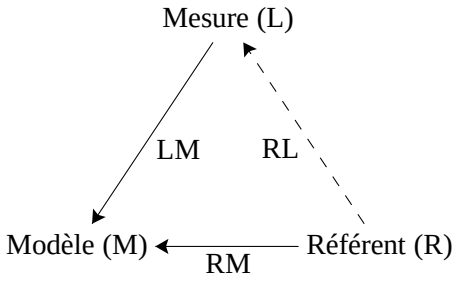


FIG. 2.22 – Le triangle de la modélisation virtuelle ou triangle sémiotique virtuel.

sujet *S* par référence en utilisant un mot qui le désigne. Par exemple, parlant d'une chaise qu'il voit, le locuteur prononce le mot "chaise", qui est alors le signifiant faisant référence à l'objet chaise, à l'adresse du sujet.

– *LS :* Le locuteur *L* évoque, à l'aide de mots, un objet invisible (il importe peu, ici que l'objet soit effectivement invisible, tel un atome à l'œil nu, abstrait ou imaginaire), ce qui suscite chez le sujet *S* le processus de production de sens.

Si nous avons ici pris le problème du langage si cher à Saussure (le père pour certains de la sémiotique et de la linguistique), Barthes [Bar53, Bar64, Bar80] et Eco [Eco72] mettront en valeur l'omniprésence de phénomènes sémiotiques dans les faits de communication, de la photographie à la publicité en passant par l'architecture. Fayeton [Fay99, Fay01, Fay02] s'attachera quant à lui à l'utilisation de la sémiotique dans la pédagogie de l'architecture urbaine. Chez Eco, le signifiant est appelé plus généralement *signe* [Eco92]. Les signifiants, pour Eco, sont «observables et descriptibles», tangibles. Les signifiés, par contre, sont «variables selon les codes à la lumière desquels nous lisons les signifiants». Ces codes articulent notre lecture de l'environnement. Ils sont historiquement et socio-culturellement situés. C'est bien là que se pose la difficulté de leur étude : «un code, bien que conventionnellement établi, ne fait plus l'objet d'une conscience de son existence par celui qui l'utilise» [BGT75].

2.5.2 Le triangle sémiotique virtuel

En ce qui concerne une modélisation virtuelle des environnements urbains, ils s'agit bien d'établir des codes grâce auxquels nous pouvons interpréter les signifiants ou représentants pour construire des modèles virtuels, signifiés ou représentés[9]. Il s'agit ici d'une interprétation virtuelle, assimilable, ici, à un processus de cognition virtuelle. Nous pouvons alors retranscrire le triangle sémiotique de Ogden et Richards en un triangle de modélisation virtuelle (voir figure 2.22). Ainsi, les environnements virtuels sont des *signifiés* de référents qui peuvent être réels ou imaginaires. Les *signifiants* ou représentants utilisés pour les construire sont des mesures : photographies, relevé laser, position GPS …. Nous retrouvons ici les mêmes situations induites que pour le triangle sémiotique :

– *RM :* Le modèle *M* est obtenu par l'observation directe du référent *R*. Cette

[9]Pour éviter toute confusion, nous utiliserons, dans le cas de la modélisation virtuelle, les termes représentant et représenté en places de signifiant et signifié

situation décrit la modélisation directe de la tasse que j'observe. Cette situation implique par ailleurs une situation de type RS dans le triangle sémiotique. Elle correspond aux méthodes que nous avons appelées manuelles.

– *RL+LM :* Le référent *R* est mesuré par *L*, utilisée pour obtenir le modèle *M*. Cette situation se présente pour toute opération de reconstruction d'un objet réel à l'aide de mesure. Elle correspond aux méthodes basées sur les données.
– *LM :* La mesure *L* est utilisée directement pour la construction du modèle *M*. Il s'agit, par exemple, de la situation de reconstruction à partir de croquis d'architecte qui ne se réfèrent pas à des objets construits (ou pas encore). Un exemple d'une telle méthode est présenté en section 2.3.2.1.

Nous avons désigné ce triangle comme virtuel, puisqu'il ne contient pas les processus de cognition liés à l'observation de l'objet, mais les mécanismes de construction du modèle qui joue, en quelque sorte, le rôle d'une image mentale pour l'ordinateur. De même, les situations présentées plus haut visent à simuler les processus de cognition humains. Néanmoins, c'est lorsqu'un sujet observe le modèle que les processus de cognition interviennent. Le modèle ne perd pas pour autant ses références à l'objet qui représente. On peut alors intégrer le triangle sémiotique virtuel dans le triangle sémiotique d'Ogden et Richards (cf. figure 2.23). En effet, le référent *R* (l'objet étudié) et la mesure *L* (une photographie de l'objet par exemple) se retrouvent dans les deux triangles, constituant leur arête commune. Néanmoins, le troisième sommet du triangle est, dans le cas du triangle sémiotique d'Ogden et Richards, le signifié *S* (l'image mentale de l'objet), et, dans le cas du triangle sémiotique virtuel, le modèle *M* (la représentation tridimensionnelle de l'objet). Dans ce nouveau triangle apparaît ainsi une nouvelle situation *MS*. Celle-ci décrit l'interprétation d'un modèle 3D par l'observateur et la production d'un signifié à partir de celui-ci. Néanmoins, il est difficile d'observer un modèle 3D directement : ce que l'on observe, en réalité, est un rendu de celui-ci. De la même façon que l'on peut observer les objets réels à travers des photographies ou vidéos, on observe les modèles 3D de façon indirecte. C'est pourquoi nous ajoutons un nouveau sommet représentant un rendu du modèle 3D, c'est à dire une image virtuelle *V*, faisant référence à la fois au modèle 3D *M* et au référent *R* (cf. figure 2.24).

Dans cette figure, formant un graphe, chaque chemin correspond à une méthode de modélisation.

– Le chemin en rouge correspond à une opération de reconstruction d'un objet réel à partir de mesures. De telles méthodes ont été présentées dans ce chapitre en section 2.1.
– Le chemin en bleu décrit les opérations de modélisation faisant référence à un objet réel sans utilisation directe de données, ce que nous avons appelé jusqu'ici les méthodes manuelles (voir section 2.3.1).

A partir de cette figure, il est possible de caractériser certaines catégories de méthodes de modélisation grâce à leur rapport sémiotique à l'objet qu'elles décrivent. Néanmoins, nous n'avons caractérisé ici que les méthodes dites spécifiques, c'est à dire les méthodes qui visent à modéliser un objet précis.

2.5.3 Sémiotique et abstraction

L'abstraction peut être définie comme la sélection d'un ensemble d'attributs ou de propriétés, parmi un plus grand ensemble, qui caractérise un type d'objet. Il existe plusieurs types d'abstraction, et nous en retiendrons quatre que nous estimons essen-

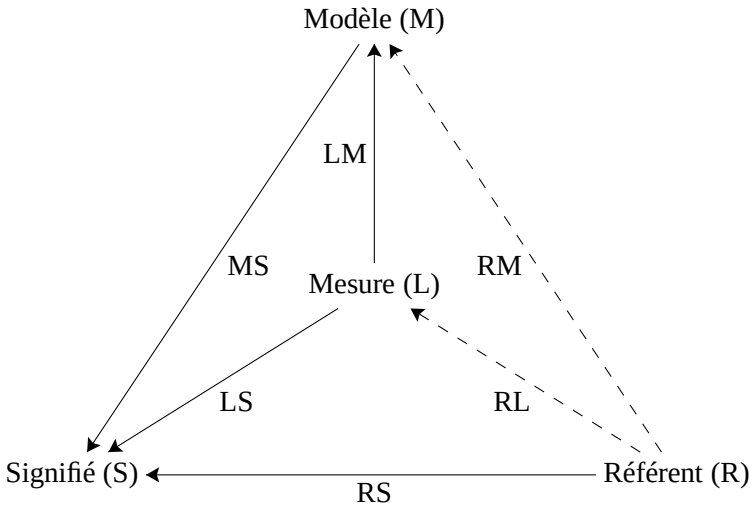


FIG. 2.23 – Le tétrahèdre sémiotique virtuel.

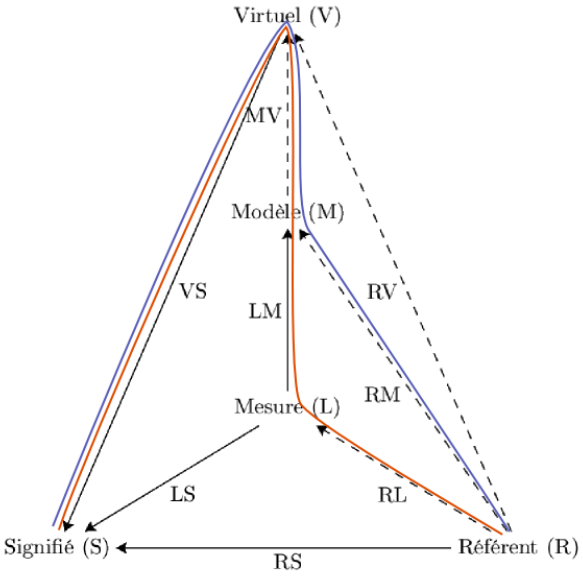


FIG. 2.24 – Intégrations du rendu dans le tétrahèdre sémiotique virtuel.

tiels. Ainsi, en étudiant des objets, on peut les classer par type ou catégories grâce à ces abstractions : c'est la *classification*. Néanmoins, parmi les objets d'une classe, on peut en trouver qui partagent plus de propriétés ou attributs que d'autres, et ainsi créer une sous-classe : c'est la *spécialisation*. À l'inverse, la *généralisation* consiste à regrouper des classes en surclasses. Ainsi, par ces processus, nous créons une *hiérarchie d'abstraction*. Par ailleurs, la *décomposition*, dont le pendant est l'*agrégation*, consiste à morceler un objet. Le *filtrage*, pour finir, est la sélection d'une partie d'une hiérarchie d'abstraction vérifiant certaines propriétés.

Nous pouvons aussi concevoir le rapport entre objet réel et mesure du point de vue de l'abstraction. En effet, un ensemble de mesures est une abstraction de l'objet qu'il représente : cet ensemble permet d'identifier certaines propriétés caractéristiques de l'objet. Plus précisément, cette abstraction particulière est un filtrage : une photographie, par exemple, capture certaines propriétés de l'objet qui dépendent du point de vue, des caractéristiques de l'appareil utilisé, de l'éclairage... De la même façon, un modèle virtuel est une abstraction de ces mesures. Il s'agit en fait d'une généralisation. Un méta-modèle[10] représente une abstraction de modèles virtuels : il s'agit aussi d'une généralisation. Pour finir, l'objet cognitif représente le haut de la hiérarchie d'abstraction ainsi définie. Néanmoins, le type d'abstraction que caractérise le rapport entre l'objet cognitif et les autres objets ici considérés ne peut être simplement classifié : il s'agit d'une combinaison de différentes abstractions.

Nous généralisons ainsi le triangle sémiotique et le triangle sémiotique virtuel comme illustré par la figure 2.25. Nous retrouvons dans la figure le triangle sémiotique d'Ogden et Richards. Nous voyons par ailleurs apparaître les méta-modèles qui encodent les règles de la cognition virtuelle. Ainsi, de nouvelles situation se présentent :

- *MC* : Le passage d'un modèle spécifique *M* à un méta-modèle *C* est ainsi une opération de généralisation ou de codification. Le passage du signifiant (la mesure *L*) au modèle spécifique *M* est une opération de reconstruction, tandis que pour passer au méta-modèle, il s'agit d'une reconstruction généralisée. La reconstruction généralisée correspond à un problème de reconstruction dont l'objectif n'est pas de reconstruire un modèle spécifique, mais les codes d'un méta-modèle. Un exemple d'une telle utilisation consisterait à inférer un méta-modèle à partir d'un ensemble de modèles 3D existants.
- *LC* : De même que la relation précédente, il s'agit d'une généralisation. Un exemple d'utilisation d'une généralisation de cette nature revient à déterminer un méta-modèle de colonnes toscanes à partir de photographies et de mesures d'un ensemble de colonnes.
- *CS* : La relation liant le méta-modèle *C* au signifié *S* est complexe. En effet, les codes implémentés par le méta-modèle tendent à représenter une partie de la connaissance sur laquelle se base l'interprétation faite par l'observateur de l'objet référent *R*.
- *RL+LC+CM+MV+VS* : Cette situation correspond aux méthodes *basées sur la représentation de connaissances* (voir section 2.4). En effet, ces méthodes visent à construire un modèle spécifique *M* en utilisant des mesures *L* de l'objet référent *R* en utilisant des connaissances préalables sur l'objet étudié. Les

[10] Par *méta-modèle*, nous entendons parler de modèle de modèles, c'est à dire de modèles qui encodent, dans une certaine mesure, des connaissances sur ces modèles et permettent de les généraliser. On pourra penser, par exemple, à des modèles procéduraux.

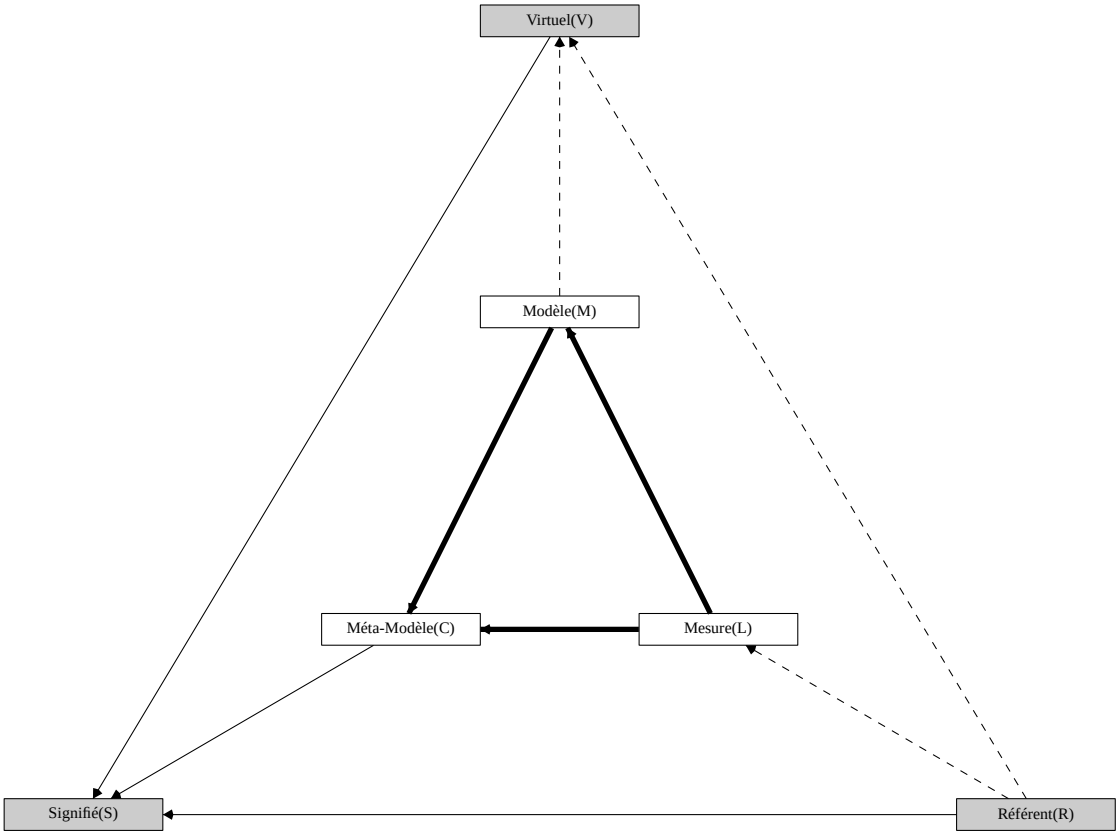


FIG. 2.25 – Le triangle sémiotique et le triangle sémiotique virtuels généralisés. Les pointillés indiquent les relations de référence, les lignes les plus épaisses les relations de généralisation.

connaissances sont alors considérées comme un méta-modèle *C* de modèles pouvant être reconstruits. À l'inverse de la relation *MC* précédemment mentionnée, le méta-modèle est ici utilisé pour créer un modèle : il s'agit alors d'une relation de spécialisation. Ces méthodes sont généralement appelées *déclaratives* puisqu'elles permettent de modéliser des objets à l'aide d'un ensemble de données sous la forme de mesures ou de propriétés sur les objets étudiés.

Nous avons ici traité des situations impliquant la modélisation d'objets référents, réels ou imaginaires. Néanmoins, ces même situations se retrouvent dans un contexte différent : celui de la modélisation comme processus de conception, c'est à dire d'objets en cours d'élaboration. Dans ce contexte, en effet, le rapport entre le signifié (l'image mentale de l'objet à modéliser), le modèle (qui est signifiant de cet objet), et le référent (la réalisation éventuelle du projet) est inversé. La conception consiste à passer d'un signifié à un référent à venir : c'est un problème de communication.

2.5.4 Sémiotique et communication

Nous jouissons de l'architecture comme d'un fait de communication, même sans en exclure la fonctionnalité.[. . .] L'architecture est une rhétorique.

Umberto Eco, La structure absente *[Eco72]*

Grumbach [Gru03] présente un état de l'art de la cognition virtuelle, c'est à dire des implications cognitives de la cognition sur le virtuel et du virtuel sur la cognition. Il traite principalement du virtuel comme d'un monde à la fois perçu (*i.e.* objet de cognition) et présentant des possibilités d'interaction. La modélisation est, en pratique, une interaction entre un signifié et un modèle. Donikian [Don04] présente un état de l'art très complet sur le sujet de la cognition virtuelle et la modélisation du comportement d'agents virtuels autonomes. De tels agents ont été utilisés pour assister un processus de modélisation d'objets architecturaux en observant les déplacements des agents dans les espaces modélisés [KE96, EK96]. Thomas [Tho05], quant à lui, propose un modèle de cartes cognitives spatiales et un modèle de mémoire pour améliorer ces comportements. Néanmoins, ces approches ne traitent pas du virtuel comme objet de modélisation ou de conception mais de perception, de compréhension de l'interaction.

Avant de construire leurs cabanes, nos pères en ont d'abord conçu l'image

Boullée, cité par Boudon dans Figuration graphique en architecture *[BGT75]*

Dans le cas de la conception d'un objet en utilisant des modèles virtuels, les relations du triangle sont inversées comme le montre la figure 2.26. En effet, le point de départ est alors l'image, le cognitif, et l'objectif est soit un objet réel, qui sera réalisé à l'issue de l'opération de conception (situation *SR*)[11], soit un signifiant en référence à l'objet à construire (situation *SM+MR*), soit un modèle purement virtuel (situation *SM+MV*). Ce modèle n'a pas de vocation généralisante sur les sciences de la conception, et a pour objectif de formaliser la création d'un modèle virtuel. Ainsi,

[11] le passage entre signifié et référent peut être identifié à la notion d'embrayage de Boudon [Bou03], tandis que le passage inverse se rapporte à la notion d'échelle

nous nous intéresserons surtout au cas où l'opération de conception tend à la création d'un modèle virtuel.

L'édifice construit est représentation d'un projet qui l'a précédé.

Boudon [BGT75]

Les situations qui nous intéressent sont donc les suivantes :

- *SC+CM+MV* : Ici, le méta-modèle *C* est utilisé en spécialisation pour générer un modèle spécifique *M*. Néanmoins, ici, aucune mesure n'entre en jeu. De même, la relation à l'objet référent *R* est éventuellement perdue. Il s'agit donc de la création d'un objet représentatif du méta-modèle. Ce type de situation correspond aux méthodes dites *procédurales*. À partir de l'image (ou signifié *S*) qu'il a de son projet, l'utilisateur va ainsi pouvoir coder un méta-modèle *C* qu'il pourra par la suite spécialiser en un modèle spécifique *M* à l'aide de paramètres.
- *SM+MV* : Il pourra, sinon directement implémenter le modèle spécifique *M*. Cette situation correspond au cas le plus courant de l'utilisation des techniques de modélisation à l'aide d'outils du commerce.
- *MV+VR* et *ML+LR* : Le modèle réalisé *M* peut servir à créer le référent *R* objectifs du projet. Soit en passant par des rendus virtuels *V*(des plans, coupes, élévations ou axonométries par exemple), ou des mesures *L* (dimensions, proportions ou cotes par exemple). Il existe alors une boucle que nous ne représentons pas ici entre les modèles *M* ainsi produits, les représentants qui pourront en être produits (rendu *V*, maquettes ou mesures *L*) et un nouveau processus de cognition sur ce représentant qui devient ainsi signifiant pour éventuellement modifier le projet.

L'homme apprend en voyant, et ce qu'il apprend retentit à son tour sur ce qu'il voit.

James J. Gibson, The perception of the visual world *[Gib74]*

Ce modèle, comme nous avons pu le voir, permet de classer les opérations de modélisation et de conception grâce à leurs relations vis à vis d'un projet, objectif, ou objet cognitif et d'un objet réel existant ou à construire. Ceci nous permet d'établir une brève typologie des méthodes existantes.

2.5.5 Une typologie des principales méthodes de modélisation

Tout d'abord, nous distinguons les *modèles spécifiques*[12] représentant une instance particulière d'un objet, et les *méta-modèles* qui représentent une famille ou une classe de modèles. Les premiers ont comme avantage la simplicité de mise en œuvre, notamment à l'aide d'outils de modélisation. Les seconds permettent de nombreuses utilisations du même modèle à plusieurs fins grâce à l'utilisation de paramètres. Par ailleurs, nous identifions deux principaux paradigmes de modélisation : la modélisation basée modèle et la modélisation basée données. Tandis que la modélisation basée

[12]Nous avons gardé l'appellation de Hoffmann [HJA02] pour les modèles spécifiques. Néanmoins, nous avons ici englobé les approches paramétriques et variationnelles sous l'appellation de méta-modèles

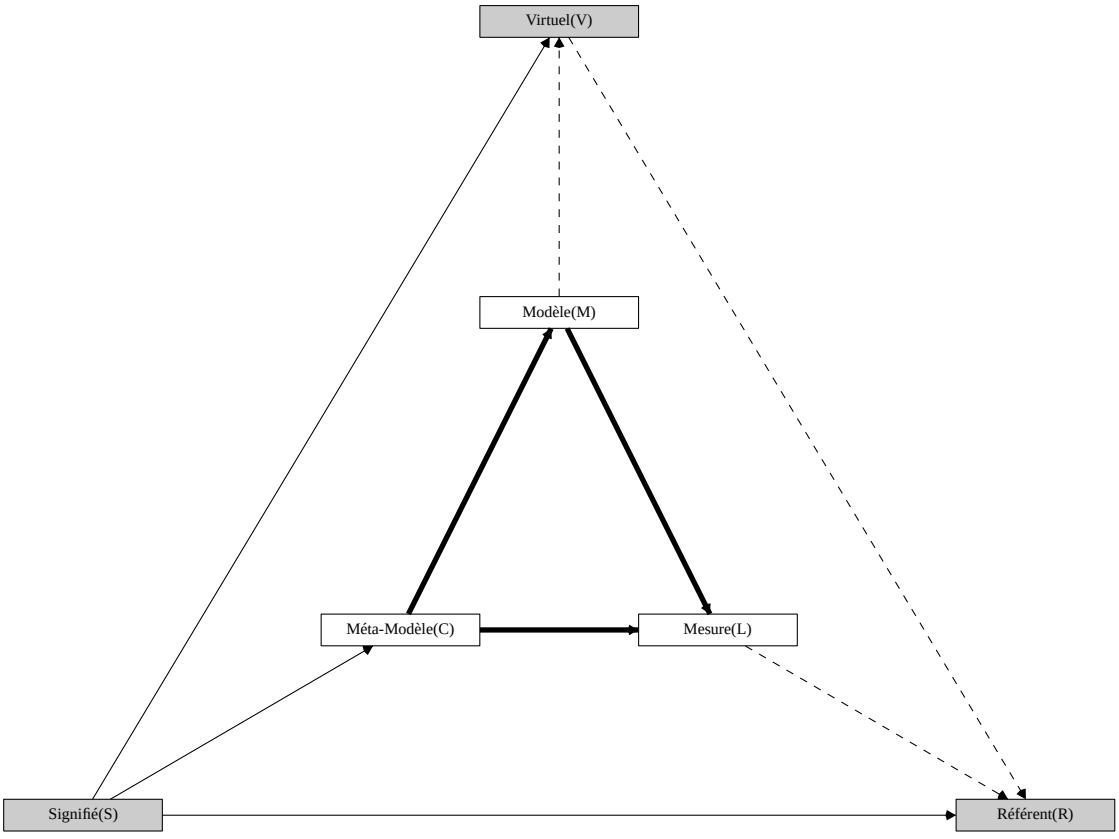


FIG. 2.26 – Le triangle sémiotique généralisé du point de vue de la conception.

modèle place le modèle au centre du processus de conception, la modélisation basée données utilise des mesures en référence à l'objet étudié.

La combinaison de deux types de modèles et de deux paradigmes de méthodes nous permet de définir quatre principaux types de méthodes de modélisation :

- Les méthodes manuelles visent à la construction directe de modèles spécifiques.
- Les méthodes basées sur les données utilisent des mesures pour reconstruire des modèles spécifiques.
- Les méthodes procédurales utilisent des méta-modèles pour créer une variété de modèles spécifiques.
- Les méthodes déclaratives utilisent des mesures pour produire des modèles spécifiques à partir de méta-modèles.

Par ailleurs, il existe de nombreuses méthodes hybrides combinant certains aspects de ces quatres méthodes de base. En effet, la combinaison de l'approche basée modèle et de l'utilisation de données permet à l'utilisateur de contrôler au fur et à mesure le processus de construction du modèle. Par ailleurs, il existe des méthodes prenant en compte des connaissances a priori sur le modèle : ces méthodes représentent un compromis entre l'utilisation de modèles et de méta-modèles puisqu'elle prennent en compte un ensemble de règles contextuelles comme contraintes. La figure 2.27 reprend les différentes méthodes présentées dans ce chapitre et les classifier en rapport au type de modèle et au paradigme qu'elles utilisent. La colonne *contexte* indique l'utilisation de règles propres au domaine d'application.

2.6 Discussion

Les environnements urbains étant des objets complexes, il est nécessaire de considérer des connaissances propres à ces environnements pour les modéliser. En effet, l'introduction de règles contextuelles permet de simplifier la modélisation et de s'attarder sur les caractéristiques les plus importantes et d'ignorer les autres. Une telle simplification peut être vue comme un mécanisme d'abstraction. Dans ce chapitre, nous avons parcouru les différentes méthodes de modélisation de la ville et étudié les mécanismes d'abstraction en jeu dans ce processus pour en dégager une typologie. Comme le montre la figure 2.27, il existe des plus en plus de méthodes combinant les avantages de l'utilisation de données et des techniques basée modèle. Par ailleurs, nous avons identifié de nombreuses méthodes utilisant des connaissances spécifiques pour franchir les obstacles provenant de l'utilisation de données. Ces méthodes sont, en effet, de plus en plus efficaces, et l'intégration de connaissances est probablement la direction la plus encourageante pour la modélisation d'environnements urbains.

Le chapitre suivant présente les méthodes de modélisation procédurales. Ces dernières induisent l'existence d'une représentation (cognitive ou non) des codes des modèles pouvant être générés. Ainsi, les méthodes procédurales utilisent des méta-modèles. Nous les traitons séparement d'une part pour leur valeur de généralisation et d'autre part pour leur expressivité qui ouvre des perspectives intéressantes dans le cadre de la modélisation urbaine.

Référence	modèle	données	spécifique	contexte	méta-modèle
[ZS01, DTC00, MZ]		×	×		
[FZ03]		×	×		
[LPC+00]		×	×		
[vVDBB98]	×	×	×		
[MBST01]	×	×	×		
[Sin03]		×		×	
[HBS+99]		×		×	
[LML+00]		×		×	
[JDPM]		×		×	
[HB99]		×		×	
[HYN03]		×		×	
[PKVG]		×		×	
[DTM96]	×	×		×	
[Rec05]	×	×		×	
[DOCJ91]	×	×		×	
[JLF96, JM99]	×	×		×	
[BJPW99]	×			×	
[Don97, Tho99, Don04]	×			×	
[Mah95]		×			×
[FBD97, Bla95, FBD00]		×			×
[Bou95, MZ94, MZ96]		×			×
[DGdL+03]		×			×
[PABCV00]		×			×
[Fau01]		×			×
[Aut01]		×			×
[Don92]	×	×			×
[Liè97]	×	×			×

FIG. 2.27 – méthodes hybrides.

Chapitre 3

Méthodes de modélisation procédurale

En informatique graphique, il est courant de vouloir réutiliser des objets géométriques déjà définis ailleurs, pour réduire l'espace de stockage d'une part, et le coût d'affichage de ces modèles lors de la navigation d'autre part. L'espace de stockage est réduit car il suffit de faire référence à un objet existant pour en placer une copie (une instance) ailleurs dans la scène. Ceci revient, dans un texte, à citer le chapitre 1 au lieu de le recopier en entier lorsqu'on y fait référence. C'est en fait une compression des données géométriques. Le coût du rendu est de même amélioré grâce aux capacités des cartes graphiques qui permettent de multiples utilisations des objets géométriques.

Nous avons ici parlé de réutilisation d'objets identiques. Néanmoins, certains objets peuvent être regroupés de par leur similitudes, comme des brins d'herbe ou les branches d'un arbre par exemple. Il ne s'agit plus ici de compression, mais d'abstraction. Nous pouvons alors définir une classe d'objets en décrivant les propriétés communes de ces objets. Les propriétés variant d'un objet à l'autre deviennent les attributs ou paramètres de la classe. Une simple méthode ou description peut ainsi servir a décrire une grande variété d'objets. C'est ce que Smith [Smi84] appelle l'*amplification des données*. Cette propriété caractérise les méthodes de modélisation procédurales que nous présentons dans ce chapitre. Nous donnons tout d'abord, en section 3.1, une définition de ces méthodes. La section 3.2 présente la paradigme d'instanciation géométrique procédurale. La section 3.3 présente, quant à elle, les méthodes de modélisation du développement, et notamment, les systèmes de Lindenmayer. Nous revenons dans le vif du sujet de ce document en section 3.4 pour traiter de la modélisation procédurale des environnements urbains. L'introduction du mouvement dans les méthodes procédurale est présentée en section 3.5, suivie des méthodes de modélisation de l'évolution en section 3.6. Nous terminons ce chapitre par une discussion sur les méthodes procédurales en section 3.7.

3.1 Une définition de la modélisation procédurale

Il existe de nombreuses définitions de la modélisation procédurale dans la littérature. Ebert [EMP+03] définit l'adjectif procédural comme qualité des entités décrites par des programmes plutôt que par des structures de données. Cependant, comme le remarque Perbet [Per03], cette distinction n'est pas satisfaisante tant qu'une définition précise des limites entre programmes et données n'est pas établie. En effet, les langages de modélisation géométriques permettent la définition, par l'utilisateur, de scripts qui peuvent créer et modifier le modèle. Ces scripts font partie du processus de modélisation et sont néanmoins bien des programmes. Cette définition étant ainsi trop floue, Perbet en propose la suivante en définissant tout d'abord la complexité d'un langage descriptif comme la taille relative de sa sémantique par rapport à celle du programme qui l'utilise. Une modélisation est alors procédurale, pour Perbet, lorsque cette complexité du langage descriptif utilisé est grande. Cette définition pose ainsi une mesure de procéduralité des méthodes en fonction de la complexité du langage descriptif qu'elles utilisent. Néanmoins, elle ne nous semble pas satisfaisante puisqu'elle ne permet pas, par exemple, de faire la distinction entre méthodes procédurales et déclaratives.

Nous utiliserons ici pour définir les méthodes procédurales la propriété mise en valeur par Smith [Smi84] : une méthode est procédurale si elle peut être assimilée à une amplification de ses données d'entrée, programmes ou paramètres. Cette

```
define  grass(0)   blade                 end
define  grass(n)
                   grass(n − 1)
                   grass(n − 1)          translate 2^n(0.1,0.0,0.0)
                   grass(n − 1)          translate 2^n(0.1,0.0,0.0)
                   grass(n − 1)          translate 2^n(0.1,0.0,0.0)
end
grass(15)
```

FIG. 3.1 – Exemple d'utilisation du paradigme d'instanciation géométrique. La première règle produit un brin d'herbe (blade). La seconde décompose une instance de $grass(n)$ en quatre instances de $grass(n-1)$ [Har92, Har94].

propriété stipule que de telles méthodes produisent, à partir d'un certain volume de données, un plus grand volume de données. Cette définition est donc plus précise et permet bien de distinguer méthodes procédurales et déclaratives tout en conservant les nuances apportées par Perbet.

3.2 Instantiation géométrique procédurale

Pour ce faire, Hart [Har92, Har94] propose le paradigme d'*instanciation d'objets géométriques* (Procedural Geometric Instancing ou PGI) dans le cadre de la modélisation d'objets fractals. Ainsi, à partir de règles simples, il devient possible de réutiliser le même arbre pour définir une forêt en comportant des centaines ; chaque arbre pouvant lui-même utiliser plusieurs fois la même branche et ainsi de suite. Ce paradigme est aujourd'hui intégré à la plupart des langages de représentation géométrique (*e.g.* VRML[MCB97]). Ce paradigme est illustré dans la figure 3.1 par un exemple. Dans celui-ci, chaque instance de $grass(i)$ produit par induction 4^i brins (blades) d´herbe.

Dans le cadre de la modélisation d'environnements naturels, Deussen [DHL+98] propose l'*instanciation approximative* comme une extension du paradigme d'instanciation de Hart. Les objets modélisés procéduralement sont projetés dans l'espace des paramètres de l'axiome et ensuite regroupés par *clusters*. Chaque *cluster* représente alors une instance approximative de tous les objets qui en font partie, réduisant ainsi le nombre d'objets que contient la scène.

3.3 Modélisation procédurale du développement : les systèmes de Lindenmayer

Lindenmayer [Lin68] introduit la notion de *L-systems* (pour Lindenmayer-systems) comme une théorie mathématique du développement des plantes. Cette théorie diffère du mécanisme de réécriture des grammaires de Chomsky [Cho63] dans la méthode d'application des productions : en effet, les productions sont appliquées séquentiellement dans les grammaires de Chomsky alors qu'elles sont appliquées en parallèle dans les L-systems. Cette différence reflète la motivation biologique des L-systems. L'application des règles de production a un impact essentiel sur les propriétés formelles d'un système de réécriture ; les langages décrits par les L-systems et les grammaires de Chomsky sont donc différents. Il existe par exemple des langages générés par les L-systems non-contextuels (dits 0L-systems) mais qui ne peuvent pas être générés par les grammaires de Chomsky non-contextuelles.

Le concept central des L-systems est un mécanisme de réécriture parallèle de chaînes de caractères basé sur un jeu de règles de production. En général, la réécriture est utilisée pour définir des objets complexes par remplacement successif de parties d'un objet initial simple. L'exemple classique est *La courbe du flocon de neige*, proposée par Von Koch en 1905 et illustrée sur la figure 3.2.

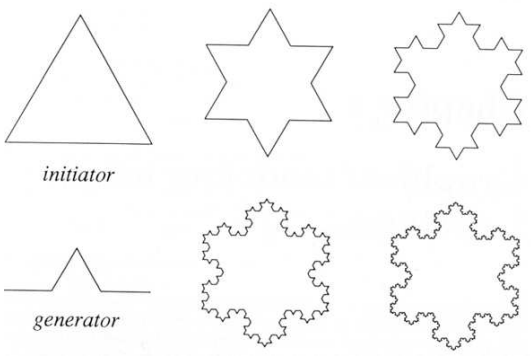


FIG. 3.2 – Construction du flocon de neige. A chaque itération, les segments sont remplacés par le générateur. [PLH+90]

A l'origine, la théorie des L-systems ne contenait pas assez de détails pour permettre de modéliser de façon suffisamment fine le comportement des plantes. De nombreux chercheurs se penchèrent sur la question, notamment P. Prunsinkievicz [PLH+90], qui a conçu une interprétation géométrique des L-systems basée sur la géométrie "tortue". Mais revenons tout d'abord à une définition formelle des L-systems :

3.3.1 Définition

Définition 1 (0L-system) *Soit V un alphabet, V^* l'ensemble des mots de V, V^+ l'ensemble non nul des mots de V. Un 0L-system est un triplet $G =< V, w, P >$, où V est l'alphabet du système, $w \in V^+$ un mot non nul dit* axiome *et $P \subset V \times V^*$ est un ensemble de productions.*
Une production $(a, \chi) \in P$ s'écrit $a \rightarrow \chi$. La lettre a et le mot χ sont respectivement le prédécesseur *et le* successeur *de cette production. Nous admettons que pour toute lettre $a \in V$, il existe au moins un mot χ tel que $a \rightarrow \chi$. Si aucune production n'est explicitement spécifiée pour un prédécesseur donné $a \in V$, nous admettons l'appartenance de la* production identité *$a \rightarrow a$ à l'ensemble des productions P. Un 0L-system est* déterministe *(on le dit alors D0L-system) si et seulement si pour chaque $a \in V$ il existe exactement un mot $\chi \in V^+$ tel que $a \rightarrow \chi$.*

Définition 2 (Dérivation) *Soit $\mu = a_1 \dots a_m$ un mot quelconque de V^*. Le mot $\nu = \chi_1 \dots \chi_m \in V^*$ est* directement dérivé *de (ou* généré par*) μ, noté $\mu \Rightarrow \nu$, si et seulement si $a_i \rightarrow \chi_i$ pour tout $i = 1, \dots, m$. Un mot ν est généré par G dans une dérivation de longueur n s'il existe une* séquence de développement *de mots μ_0,μ_1,...,μ_n tels que $\mu_0 = \omega$, $\mu_n = \nu$ et $\mu_0 \Rightarrow \mu_1 \dots \Rightarrow \mu_n$.*

Exemple 1 *L'exemple suivant utilise ce formalisme pour décrire le développement d'un fragment de filament multicellulaire d'*Anabaena *[Pru03]. Cet organisme divise ses activités de photosynthèse et de fixation de l'azote spatialement en deux types de cellules notées L et S qui sont par ailleurs polarisées, ce qui est indiqué par les*

flèches. Le système peut être décrit par le L-system suivant :
ω : $\overrightarrow{L}$
$p_1 : \overrightarrow{S} \rightarrow \overrightarrow{L}$
$p_2 : \overleftarrow{S} \rightarrow \overleftarrow{L}$
$p_3 : \overrightarrow{L} \rightarrow \overleftarrow{L}\,\overrightarrow{S}$
$p_4 : \overleftarrow{L} \rightarrow \overleftarrow{S}\,\overrightarrow{L}$

En partant d'une simple cellule $\overrightarrow{L}$ (l'axiome), la figure 3.3 représente la séquence de mots générée.

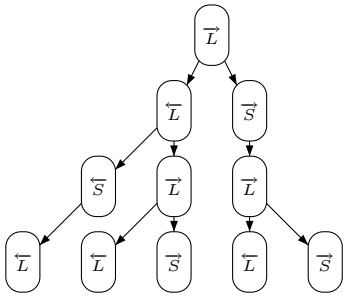


FIG. 3.3 – Développement d'un fragment de filament multicellulaire. [Pru03]

Néanmoins, le précédent exemple ne prend pas en compte les différentes durées des développements décrits. En effet, la division des petites cellules S (règles $p1$ et $p2$) est en réalité 20% plus longue que celle des longues cellules L (règles $p3$ et $p4$) [Pru03]. Une solution consisterait à introduire des règles intermédiaires pour simuler ce phénomène, mais ceci ne constitue pas une solution acceptable dans certains cas, notamment si ces différentiels de développements ne sont pas rationnels ou varient. Une solution plus appropriée consiste à affecter des paramètres aux symboles du L-system.

3.3.2 L-systems paramétrés

Définition 3 (Mot paramétré) *Un L-system paramétré opère sur des* mots paramétrés*, qui sont des chaînes de* modules*. Un* module *est constitué de* lettres *(formant son nom) et de* paramètres formels *associés. Les lettres appartiennent à un alphabet V et les paramètres formels à l'ensemble des nombres réels $\mathcal{R}$. Un module, de nom $A \in V$ et dont les paramètres formels sont $a_1, \ldots, a_n \in \mathcal{R}$, est dénoté par $A(a_1, \ldots, a_n)$. Tout module appartient à l'ensemble $M = V \times \mathcal{R}^*$, où $\mathcal{R}^*$ est l'ensemble des séquences finies de paramètres formels. L'ensemble de toutes les chaînes de modules et l'ensemble des chaînes non vides sont dénotés par $M^* = (V \times \mathcal{R}^*)^*$ et $M^+ = (V \times \mathcal{R}^*)^+$ respectivement.*

Définition 4 (Expression) *Si Σ est un ensemble de* paramètres formels*, alors $\mathcal{C}(\Sigma)$ et $\mathcal{E}(\Sigma)$ dénotent respectivement une* expression logique *et une* expression arithmétique *de paramètres formels de Σ.*

Définition 5 (L-system paramétré) *Un* L-system paramétré *se définit alors comme un quadruplet ordonné $G =< V, \Sigma, \omega, P >$ où :*

- *V est l'alphabet du système.*
- *Σ est l'ensemble des paramètres formels.*
- *$\omega \in (V \times \mathcal{R}^*)^+$ est un mot paramétré non vide dit axiome. Dans l'axiome, tous les paramètres formels sont affectés (ce sont alors les paramètres du mot paramétré).*
- *$P \subset (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times \mathcal{E}(\Sigma))^*$ est un ensemble fini de productions. $(V \times \Sigma^*)$ est l'ensemble des prédécesseurs de la production, $\mathcal{C}(\Sigma)$ l'ensemble des conditions, et $(V \times \mathcal{E}(\Sigma))^*$ l'ensemble des successeurs de la production.*

Les symboles " :" et "$\rightarrow$" séparent les trois composantes de la production : le prédécesseur, *la* condition, *et le* successeur. *Par exemple, une production dont le prédécesseur est $A(t)$, la condition $t > 5$, et le successeur $B(t+1)$ s'écrit comme suit :*

$$A(t) : t > 5 \rightarrow B(t+1)$$

Elle se lit : $A(t)$ se dérive en (devient) $B(t+1)$ si $t > 5$.

Définition 6 (Dérivation conditionnelle) *Une production* satisfait *un module dans un mot paramétré si et seulement si les conditions suivantes sont vérifiées :*

- *la lettre du module correspond au prédécesseur de la production.*
- *le nombre de paramètres du module est équivalent à celui des paramètres formels du prédécesseurs de la production.*
- *la condition appliquée aux paramètres formels de la production est évaluée à* vrai.

Si un module a produit un mot paramétré χ comme résultat de son *application* à un L-system G, nous écrivons $a \mapsto \chi$. Étant donné un mot paramétré $\mu = a_1 \dots a_m$, nous disons que le mot $\nu = \chi_1 \dots \chi_m$ est *directement dérivé* de (ou *généré* par) μ et nous écrivons $\mu \Longrightarrow \nu$ si et seulement si $a_i \mapsto \chi_i$ pour tout $i = 1, \dots, m$. Un mot paramétré ν est généré par G selon une *dérivation de longueur n* s'il existe une séquence de mots $\mu_0, \dots, \mu_n$ telle que $\mu_0 = \omega$, $\mu_n = \nu$ et $\mu_0 \Longrightarrow \mu_1 \dots \Longrightarrow \mu_n$.

Exemple 2 *Reprenons l'exemple 1. Ici, un seul module M est utilisé pour les deux types de cellules L et S, mais les paramètres t et p représentent le développement et la polarité des cellules. Ainsi, en attribuant un développement de départ de 1 pour les cellules S, de 2 pour les cellules L et un seuil de division des cellules de 5, un L-system équivalent à celui de l'exemple 1 est obtenu en y intégrant un différentiel dans les durées de division des deux types de cellules [Pru03].*
ω : $\boldsymbol{M}(5, RIGHT)$
$\boldsymbol{M}(t,p) : t < 5 \rightarrow \boldsymbol{M}(t+1, p)$
$\boldsymbol{M}(t,p) : t == 5 \&\& p == LEFT \rightarrow \boldsymbol{M}(1, LEFT) \quad \boldsymbol{M}(2, RIGHT)$
$\boldsymbol{M}(t,p) : t == 5 \&\& p == RIGHT \rightarrow \boldsymbol{M}(2, LEFT) \quad \boldsymbol{M}(1, RIGHT)$

3.3.3 Le développement en temps continu et les règles de décomposition

Le langage ainsi défini permet de modéliser de nombreux phénomènes de développement d'organismes biologiques. Par contre, afin d'étudier avec plus de finesse ces développement et de réaliser des animations, le contrôle des paramètres est insuffisant. En effet, il devient alors nécessaire de pouvoir définir des pas de temps

arbitraires. Les mécanismes décrits ne suffisent donc plus puisqu'un pas de temps trop grand induit des erreurs dans la division des cellules. De nouvelles règles sont ainsi introduites dans le langage par Prusinkiewicz [PMKL01a]. Ces règles, appelées règles de décomposition, sont appliquées récursivement, à la manière de règles de Chomsky [PMKL01b]. Elles ont la même forme que les autres règles, mais sont précédées du mot clef *decomposition*. Ainsi, chaque étape de dérivation parallèle est suivi de l'application récursive des règles de décomposition.

Exemple 3 *L'exemple 2 devient ainsi :*
ω : $\boldsymbol{M}(0, RIGHT)$
$\boldsymbol{M}(t,p) :\rightarrow \boldsymbol{M}(t+dt,p)$
$decomposition$
$\boldsymbol{M}(t,p) : t >= t_{div} \&\& p == LEFT \rightarrow \boldsymbol{M}(t - t_{div} + t_s, LEFT) \quad \boldsymbol{M}(t - t_{div} + t_l, RIGHT)$
$\boldsymbol{M}(t,p) : t >= t_{div} \&\& p == RIGHT \rightarrow \boldsymbol{M}(t - t_{div} + t_l, RIGHT) \quad \boldsymbol{M}(t - t_{div} + t_s, RIGHT)$
Dans cet exemple, nous avons :

- dt *est le pas de temps entre deux étapes de réécriture parallèle,*
- t_{div} *représente la temps que mettent les cellules pour se diviser,*
- t_s *et* t_l *sont symbolisent les âges respectifs des cellules courtes et longues après la division.*

Ici, ce sont les paramètres t_s *et* t_l *qui régulent les différentes durées de développements des cellules courtes et longues : après la division d'une cellule, la cellule longue produite est vieillie d'une durée* t_l*. Elle mettra ainsi moins de temps à se diviser comme l'illustre la figure 3.4.*

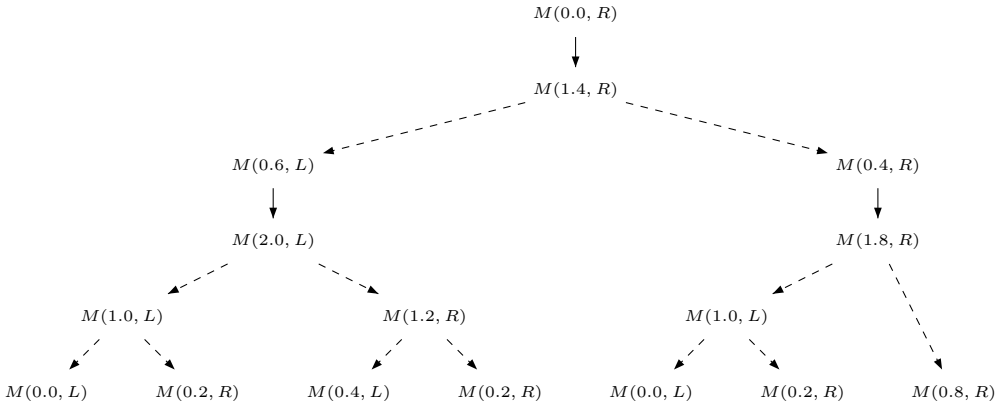


FIG. 3.4 – Développement d'un fragment de filament multicellulaire en temps continu à l'aide de l'exemple 2, avec les paramètre $dt = 1.4$, $t_{div} = 1.0$, $t_s = 0$, et $t_l = 0.2$. Les productions temporelles sont marquées par des flèches simples, les règles de décomposition par des pointillés [Pru03]. On remarque que le temps de division des cellules courtes est bien 20% plus long pour celui des cellules longues.

Ce modèle gère donc correctement des pas de temps adaptés à l'animation du développement, et permet d'obtenir l'état du modèle à un temps t quelconque. Il est ainsi plus souple puisque ce n'est plus le nombre d'étapes de réécriture qui est spécifié.

3.3.4 Une interprétation "tortue" des L-systems

Afin de visualiser les modèles générés, Prusinkiewicz [PLH+90] présente une interprétation des L-systems basée sur une tortue inspirée de *LOGO* [Ad81]. Elle

a été beaucoup utilisée depuis dans la modélisation réaliste de plantes [DHL+98], la description de motifs de *kolam* (art sud-indien) [Fer99], la synthèse de partitions musicales ...

Définition 7 (Interprétation Tortue) *Un* état *de la tortue est défini par un triplet* (x, y, α)*, où les coordonnées cartésiennes* (x, y) *représentent la* position *de la tortue, et l'angle* α *est la direction dans laquelle regarde la tortue. Étant donné un* pas d *et un* incrément angulaire δ*, la tortue répond aux commandes représentées par les symboles suivants :*

F	*Avancer d'un pas de longueur* d*. L'état de la tortue devient* $(x + d\cos\alpha, y + d\sin\alpha, \alpha)$*. Une ligne est tracée entre l'ancien et le nouvel état de la tortue.*
f	*Avancer d'un pas* d *sans tracer de ligne.*
+	*Tourner d'un angle* δ *. Le nouvel état de la tortue est* $(x, y, \alpha + \delta)$*.*
-	*Tourner d'un angle* δ *. Le nouvel état de la tortue est* $(x, y, \alpha - \delta)$*. L'orientation positive est le sens trigonométrique.*

Exemple 4 (Ile de Koch quadratique) *On peut par exemple obtenir des approximations de l'*île de Koch quadratique *en interprétant le L-system suivant :*

ω : $F - F - F - F$

p : $F \rightarrow F - F + F + FF - F - F + F$

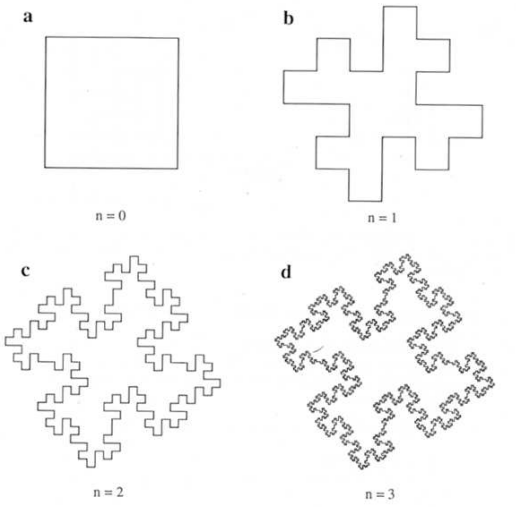


FIG. 3.5 – Génération d'une île de Koch quadratique. [PLH+90]

la figure 3.5 illustre les dérivations de longueur 0 à 3 obtenues. L'incrément angulaire δ est de 90°.

Il est alors simple d'étendre cette interprétation à la 3ème dimension. C'est dans cette forme étendue que peuvent être définis des objets en 3 dimensions, mais quelques outils sont nécessaires à la définition des structures complexes. Par ailleurs, Kurth [Kur94] définit le concept plus général de *règles d'interprétation* dans le cadre des grammaires de croissance. Ces règles ne modifient pas la séquence de mots générée, mais constituent un homomorphisme de ces chaînes de caractères. Ceci correspond à l'affectation d'une interprétation géométrique des mots du langage, proche d'une sémantique opérationnelle. Nous ne traiterons pas ici du problème de continuité géométrique des modèles engendrés (pour plus de détails voir [PLH+90]).

3.3.5 Les branchements

Pour définir des structures à branches telles que les *arbres axiaux*, Lindenmayer définit une représentation sous forme d'arbre : les *chaînes parenthésées*. Chaque expression parenthésée correspond à une branche (un sous-arbre) de l'arbre courant. Deux nouveaux symboles sont alors définis. Ils sont assimilables à des opérations d'empilement et de dépilement de l'état du module courant :

[Empiler l'état courant.
] Dépiler l'état courant.

L'état courant d'un module comprend des données telles que la position, l'orientation, mais aussi la couleur, l'épaisseur des lignes...
La figure 3.6 illustre ce formalisme.

FIG. 3.6 – Représentation en chaîne parenthésée d'un arbre axial. [PLH+90]

Des structures d'arbres peuvent désormais être définies, mais la combinaison de ces arbres (entièrement déterministes et identiques) dans une image produirait des effets d'une régularité toute artificielle. Pour pouvoir produire des plantes de même aspect général mais contenant des variations les rendant différentes, il faut définir un développement aléatoire, probabiliste.

3.3.6 L-systems stochastiques

Définition 8 (L-system stochastique) *Un* 0L-system stochastique *est un quadruplet ordonné* $G_\pi =< V, \omega, P, \pi >$. *L'alphabet* V, *l'axiome* ω *et l'ensemble des productions* P *sont définis comme dans un 0L-system (définition 1). La fonction* $\pi : P \rightarrow (0, 1)$, *appelée* distribution de probabilité, *fait correspondre à l'ensemble des productions des* probabilités de production. *Nous admettons que pour toute lettre* $a \in V$, *la somme des probabilités de toutes les productions ayant pour prédécesseur* a *est égale à 1.*

Définition 9 (Dérivation stochastique) *Nous appelons une dérivation* $\mu \Rightarrow \nu$ dérivation stochastique *dans* G_π *si pour toute* occurrence *de la lettre* a *dans le mot* μ, *la probabilité d'appliquer la production* p *avec le prédécesseur* a *est égale à* $\pi(p)$.*Ainsi, différentes productions peuvent être appliquées à plusieurs occurrences du même prédécesseur dans un pas de dérivation.*

Exemple 5 *Dans cet exemple, les probabilités de production sont indiquées au dessus du symbole de dérivation* $\rightarrow$. *Les trois productions ont une probabilité d'approximativement* $1/3$.

ω : F
$p1$: $F \overset{0.33}{\rightarrow} F[+F]F[-F]F$
$p2$: $F \overset{0.33}{\rightarrow} F[+F]F$
$p3$: $F \overset{0.34}{\rightarrow} F[-F]F$

La figure 3.7 montre des exemples de structures générées. Notez que ces structures ressemblent à des spécimens de la même espèce.

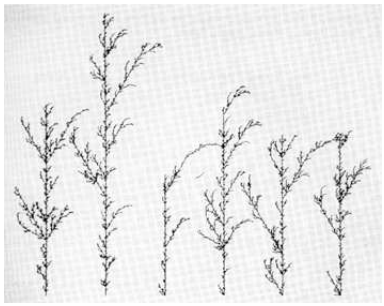


FIG. 3.7 – Structures stochastiques avec branchements. [PLH+90]

3.3.7 Modélisation procédurale de villes

Parish et al. [PM01] proposent le système *CityEngine* pour la génération procédurale de modèles de villes virtuelles. Ce système utilise un L-system hiérarchique pour la génération du réseau routier d'une ville à partir de données statistiques telles que la carte de densité de sa population ou le relevé du relief. Les règles utilisées tentent de reproduire les motifs du réseau urbain de villes réelles (concentrique pour Paris, quadrillé pour Manhattan, etc.). La génération des bâtiments est aussi faite par un second L-system à partir de volumes primitifs modifiés grâce à des règles d'extrusion, de changement d'échelle, de translation et de rotation. Les modèles générés sont intéressants et produisent un effet réaliste en survol aérien. Néanmoins, ils ne sont pas suffisamment précis pour permettre une navigation de type piéton. En effet, les bâtiments sont représentés par des géométries simples sur lesquelles sont plaquées des textures procédurales. Si cette technique convient au survol, l'absence de géométrie des façades devient visible et gène le réalisme lors d'une navigation de type piéton. De plus, ces modèles sont générés de façon statique et leur absence de structure n'autorise pas l'augmentation du détail pour leur utilisation en temps-réel.

Le système est basé sur un mécanisme de pipeline avec des cartes[1] en entrée. Différents types de cartes peuvent êtres fournis :

- Cartes géographiques :
  - Cartes d'élévation (ou de niveau) du terrain.
  - Cartes indiquant l'eau, les terres, la végétation.
- Cartes socio-statistiques :
  - Densité de population.
  - Cartes de zones (résidentielles, commerciales, mixtes).
  - Motifs de rues.
  - Cartes de hauteur (hauteur maximale d'un bâtiment).

Nous allons maintenant décrire les deux L-systems mis en place par Parish et al. [PM01]. Le premier L-system est détaillé dans les sections 3.3.7.1, 3.3.7.2 et 3.3.7.3. Le second est présenté en section 3.3.7.4.

[1] *image maps* en anglais

FIG. 3.8 – Cartes utilisées dans CityEngine [PM01]. Colonne de gauche : Eau, élévation, densité de population. A droite : Réseau routier généré par les données fournies.

3.3.7.1 Le premier L-system : création du réseau urbain

Le premier L-system, stochastique et paramétré (cf. section 1), manipule des modules qui emmagasinent les paramètres et permettent la création du réseau. Pour permettre une plus grande extensibilité, le L-system fournit à chaque pas d'itération un successeur générique dit *successeur idéal*. La modification des paramètres est alors à la charge de *fonctions externes*. On peut distinguer deux étapes de résolution dans la création du réseau urbain :

- la satisfaction des *buts globaux*
  - motifs de rues
  - densité de population
- le respect de *contraintes locales* (liées à l'environnement)
  - limites de terrains, étendues d'eau, parcs
  - élévation de terrain
  - carrefours de rues

L'axiome du premier L-system est un module représentant une route. Pour la résolution du système, nous avons alors la séquence suivante :

- Le L-system propose le *successeur idéal* du module courant (en commençant par l'axiome). Les paramètres ne sont pas affectés.
- Une fonction externe détermine tous les paramètres du successeur idéal en prenant en compte les *buts globaux*.
- Une seconde fonction externe vérifie si les paramètres conviennent aux *contraintes locales* de l'environnement. Les paramètres sont alors ajustés pour respecter ces contraintes. Si la fonction ne parvient pas à trouver de paramètres adéquats, le module sera supprimé.

Chaque module représente une route et peut engendrer la création de plusieurs autres modules, et donc de plusieurs routes.

3.3.7.2 Buts globaux

Densité de population Une recherche de Fuesser[Fue97] a mis en valeur les autoroutes comme la liaison principale entre les différentes zones à fortes densité de population d'une ville. Au contraire, les rues, se développant dans les zones résidentielles, donnent aux habitants accès à l'autoroute la plus proche. Ainsi, pendant la détermination des paramètres du successeur idéal d'un module d'autoroute, la fonction externe dédiée à cette tâche cherche le prochain centre de population : elle lance des rayons à partir de l'extrémité courante de l'autoroute et une portion d'autoroute est tracée dans la direction de plus forte densité de population (cf. figure 3.9) ; les paramètres du module sont affectés.

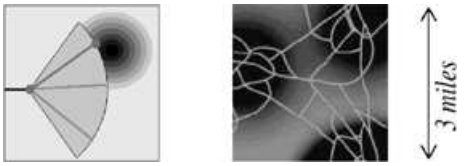


FIG. 3.9 – A gauche : L'extrémité de la route lance des rayons pour trouver un pic de population (zone sombre). A droite : Un réseau d'autoroutes reliant des centres de population. [PM01].

Motifs de rues A l'inverse, le tracé des rues n'est pas influencé par les gradients de population. Les rues suivent typiquement un motif dominant ou une combinaison de motifs.

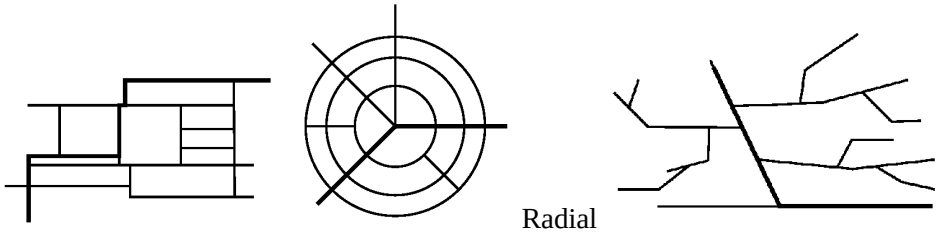


FIG. 3.10 – Sélection de motifs fréquents de rues. [PM01].

Ces motifs sont surtout appliqués pour la détermination des paramètres de route : les autoroutes ne sont affectées que par les densités de population. Les différents types de motifs sont représentés par des règles (cf. figure 3.11) :

- *Règle de base :* La règle la plus simple. Il n'y a pas de motif. Toutes les routes suivent la densité de population. Les autres règles sont des restrictions de celle-ci dont les choix d'angles et de longueurs de segments sont contraints.
- *Règle de New York (ou du jeu de dames) :* Les routes suivent un angle, une longueur et une largeur maximale de bloc. C'est le motif le plus courant dans les zones urbaines américaines, formant des blocs rectangulaires.
- *Règle de Paris (ou radiale) :* Les autoroutes suivent un tracé circulaire autour d'un ou plusieurs (motif polycentrique) centres de population.
- *Règle de San Francisco :* Les routes se conforment soit à un minimum soit à un maximum de dénivellation. Ce motif est principalement observé dans des zones au relief accentué.

Pattern name	Pattern	Example
Basic	No superimposed pattern.	
New York	Rectangular Raster	
Paris	Radial to center	
San Francisco	Elevation min or max	

FIG. 3.11 – Motifs utilisés dans *CityEngine*. [PM01].

Le système permet l'utilisation simultanée de motifs : si plusieurs motifs sont actifs au même endroit, les paramètres proposés sont combinés, permettant ainsi la création de réseaux complexes (cf. figure 3.12).

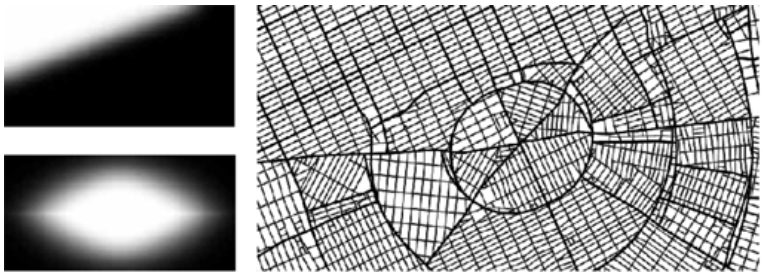


FIG. 3.12 – A gauche : Deux motifs différents (New York et Paris). A droite : Motif résultant de leur combinaison. [PM01].

On cesse de construire des routes lorsque l'on atteint une zone non peuplée.

3.3.7.3 Contraintes locales

Les fonctions externes en charge des contraintes locales ajustent les valeurs des paramètres proposées par les buts globaux pour se conformer aux propriétés de l'environnement local. Si aucune solution respectant ces contraintes n'est trouvée, le module courant est supprimé. La vérification des contraintes locales s'effectue en deux phases (chaque phase est contrôlée par une fonction externe) :

- Le segment de route est-il dans ou croise-t-il une zone illégale (eau, parc, ...) ? Le système peut tenter de réajuster les paramètres de trois manières :
  - Réduire le segment de route pour qu'il reste dans une zone autorisée.
  - Dévier le segment jusqu'à ce qu'il soit complètement dans une zone autorisée. Ce qui permet la création de routes qui longent une étendue d'eau ou un parc.
  - Les autoroutes peuvent traverser des zones illégales sous certaines conditions. La portion d'autoroute créée est marquée, et sera, au stade de création de géométrie, remplacée par un pont, un tunnel ou autre.
- Le segment de route coupe-t-il ou finit-il près d'une autre route ? En fonction de la situation, la fonction externe peut ajuster les paramètres selon les différentes corrections illustrées sur la figure 3.13 :
  - Deux rues qui se coupent forment une intersection.

– Une rue qui se termine près d'un carrefour est prolongée jusqu'à ce dernier.
– Une rue proche d'en couper une autre est prolongée pour créer une intersection.

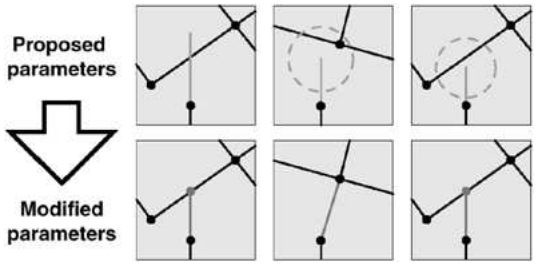


FIG. 3.13 – Exemples de contraintes locales. Ligne supérieure : paramètres proposés par les buts globaux. Ligne inférieure : paramètres corrigés. [PM01].

La figure 3.14 montre la création d'un réseau urbain basée sur des cartes de Manhattan. La partie ancienne (à gauche sur la figure) est générée en utilisant la règle de base (pas de motif), les parties récentes (à droite sur la figure) grâce à la règle New York. On peut remarquer, longeant la côte de l'île, une autoroute qui existe à la fois à Manhattan et dans sa version virtuelle. On peut aussi estimer que les positions des différents ponts sont proches des emplacement réels. Il serait néanmoins peu réaliste d'espérer obtenir un tracé vraiment similaire. En effet, les paramètres historiques et politiques (qui sont difficilement modélisables) ont une influence cruciale sur le tracé des routes.

3.3.7.4 Le deuxième L-system : Génération de géométrie - bâtiments

Division en lotissements Une fois le réseau urbain généré, le système crée des zones délimitées par les rues. Les blocs ainsi obtenus sont réduits et subdivisés en lotissements. Les lotissements trop petits ou n'ayant pas accès à la rue sont supprimés.

Le second L-system utilisé par Parish et al. [PM01] crée alors un bâtiment par lotissement. Le système gère trois types de bâtiments :
– Les gratte-ciels.
– Les commerces.
– Les résidences.

Le choix est fait grâce aux cartes de zones. Chaque type de bâtiment possède un ensemble de règles de production (ou règles de réécriture).

Génération de géométrie Un bâtiment est créé par manipulation du plan au sol du lotissement sur lequel il est construit. Les modules du L-system utilisé sont composés de transformations géométriques appliquées sur ce plan (qui est dans ce contexte un polygone convexe) : translation, changement d'échelle, extrusion, branchement, terminaison. Les toits, antennes et autres éléments architecturaux manipulés sont des objets prédéfinis. La figure 3.15 illustre le haut degré de complexité atteint par ce mécanisme très simple.

L'axiome du L-system est la boîte englobante du bâtiment, créée par extrusion du plan au sol. La sortie de chaque pas d'itération peut être interprétée comme un raffinement de la géométrie : les niveaux de détails sont générés automatiquement

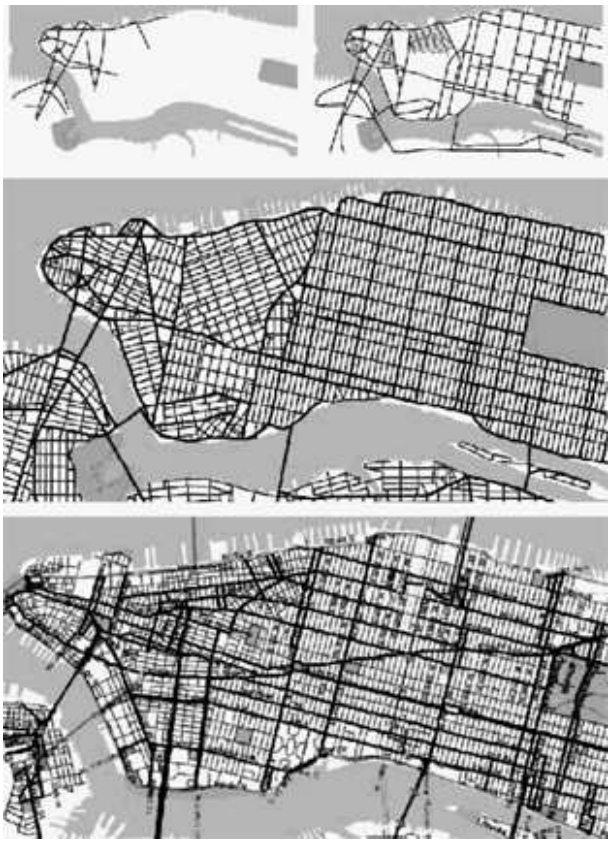


FIG. 3.14 – Création de rues appliquée à Manhattan. Ligne du haut : réseau après 28 et 142 pas d'itération. Milieu : réseau final. Bas : réseau réel de Manhattan. [PM01].

FIG. 3.15 – Cinq pas de la génération d'un bâtiment. L'axiome du L-system étant la boîte englobante du bâtiment, on a ainsi différents niveaux de détails. [PM01].

par le L-system (de par sa structure même). La figure 3.16 montre le résultat de l'intégration des bâtiments dans le réseau urbain illustré par la figure 3.14. La génération d'un tel modèle a pris plusieurs jours. De plus, le rendu n'est pas interactif.

Par une approche similaire, Marsault [MBST01] propose une génération de modèles de villes par des L-systems utilisant un vocabulaire binaire. Ces travaux sont illustrés par les figures 3.17 et 3.18.

3.3.8 Autres modèles biologiques

Il existe de nombreux autres modèles du développement. Les systèmes de *réaction-diffusion* expriment les concentrations de différentes espèces chimiques à l'aide d'équations qui spécifient les vitesses de diffusion, de réaction et de dissipation de chaque

FIG. 3.16 – Quelque part dans un Manhattan virtuel. [PM01].

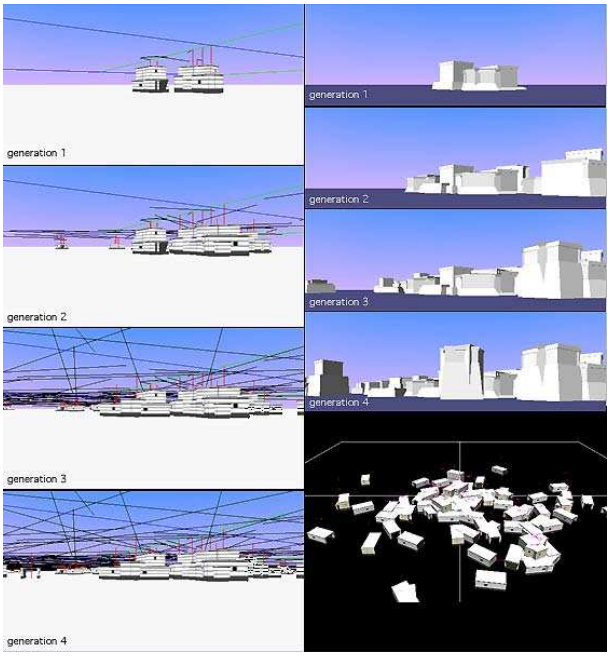


FIG. 3.17 – A partir de simples règles binaires basées sur un formalisme L-System, ce générateur récursif produit rapidement des géométries complexes.

FIG. 3.18 – Un générateur de textures pseudo-aléatoires.

espèce en jeu. En informatique graphique, les systèmes de réaction-diffusion ont été utilisés pour modéliser des textures 2D sur des surfaces [WK91, Tur91] (tissu, pierre, …). Ces systèmes sont continus dans leurs définitions. Néanmoins, ils sont souvent discrétisés spatialement sur une grille pour simplifier les calculs, souvent complexes

et longs, et produire des textures. Les *automates cellulaires* sont quant à eux discrets aussi bien temporellement que spatialement. Ils sont ainsi organisés spatialement sur une grille le plus souvent 2D ou 3D. Chaque élément de cette grille est une cellule caractérisée par un état. Les transitions des automates prennent en compte les états des cellules voisines pour déterminer le nouvel état d'une cellule. Greene [Gre89] propose l'utilisation d'automates cellulaires 3D appelés *voxel automata* (automates de pixels volumétriques) pour la croissances de plantes. Ces automates permettent notamment la prise en compte du voisinage 3D des voxels. Combaz [Com04] propose un modèle mécanique déformable pour la génération de formes.

3.3.9 Automates cellulaires, systèmes multi-agents et étalement urbain

Les travaux du laboratoire *CASA* (Centre for Advanced Spatial Analysis) à Londres proposent de nouveaux modèles d'automates cellulaires pour modéliser la croissance urbaine. En effet, une limitation commune aux automates cellulaires concerne le voisinage [OT00]. Ainsi, pour prendre en compte l'état de cellules éloignées, Engelen et al. [EWU97] utilisent un voisinage plus grand (113 cellules voisines) que le traditionnel voisinage de Van Neumann (8 cellules voisines). D'autres modèles décomposent la grille de façon irrégulière [Kau84]. Batty et Torrens [BT01] proposent une meilleure prise en compte du voisinage par l'utilisation d'agents pour effectuer les transitions. En effet, au lieu d'appliquer les règles de transition sur toutes les cellules en même temps, leur modèle déplace des agents sur la grille. Ces agents possèdent 7 règles qui leur permettent de se déplacer mais aussi de modifier l'état des cellules qu'ils rencontrent. Leur modèle montre ainsi une meilleure simulation du développement urbain. Par ailleurs, de Almeida et al. [dAMC$^+$02] proposent l'utilisation de méthodes probabilistes pour le calibrage d'automates cellulaires sur des situations réelles. Ce modèle probabiliste, qui commande des automates cellulaires, permet d'étudier les capacités de prédiction de tels modèles. Pour plus de détails, Torrens [Tor00] propose un état de l'art des automates cellulaires pour le développement urbain. Néanmoins, malgré une amélioration de la prise en compte du voisinage et de le calibrage des modèles par automates cellulaires, il subsiste un inconvénient majeur à ces méthodes : il est en effet particulièrement difficile de représenter certaines caractéristiques importantes des environnements urbains (tels que les réseaux de transports et les rivières) à l'aide de grilles.

Dans la section suivante, nous allons présenter des méthodes procéduales adaptées aux environnements urbains.

3.4 Modélisation procédurale d'environnements urbains

3.4.1 Le plan et les grammaires de forme

Les *grammaires de formes* (ou shape grammars) ont été introduites par Stiny [SG72, Sti75] et Gips [Gip74]. Ces grammaires utilisent des formes géométriques (shapes) comme alphabet. Elles sont le plus souvent planaires, mais ont été aussi appliquées au cas tridimensionnel. Utilisées initialement pour la génération de peintures [SG72, Sti75, Gip74], l'utilisation la plus répandue à ce jour de ces grammaires de formes est la génération de plans architecturaux [SM78, Fle87, Kni99, Kni00]. En effet, il est possible de représenter certaines règles de conception architecturale à l'aide de ces grammaires.

Néanmoins, une des difficultés majeures des grammaires de forme est l'application des règles. En effet, puisque le vocabulaire est constitué de formes, il faut, pour reconnaître les conditions d'application des règles, reconnaître le contexte géométrique des formes elles-mêmes. Malheureusement, ce problème n'est pas résolu, à l'heure actuelle, dans sa totalité (on pourrait même se demander s'il le sera un jour et s'il est, à vrai dire, bien formé). Ainsi, la plupart des implémentations qui effectuent la génération automatique de formes à l'aide de grammaires de formes limitent les règles et leur contexte à un sous-ensemble simple. Tapia [Tap92] s'est en effet intéressé aux panneaux chinois, et Cagan et Agarwal [Cag01, AC98] à la modélisation de machines à café. Ce qui remet en question la capacité même de ces grammaires dans le contexte de la génération automatique de plans architecturaux qui possèdent, le plus souvent, des contextes complexes : il est en effet difficile de limiter les mots générés (les plans) au langage qui nous intéresse. La solution la plus souvent adoptée est la sélection des règles par l'utilisateur.

Par ailleurs, afin d'éviter la modélisation manuelle des règles, Gero et al. proposent l'utilisation d'algorithmes génétiques pour *apprendre* les règles à partir d'exemples du langage étudié.

3.4.1.1 Les grammaires de décomposition

Wonka et al. [WWSR03] proposent un formalisme simplifié inspiré des grammaires de formes. En effet, le vocabulaire de leurs grammaires représente bien des formes. Néanmoins, ils limitent les règles à la décomposition. La figure 3.19 illustre quelques règles de décomposition : les rectangles blancs représentent les parties gauches des règles, c'est à dire les formes non-terminales du langage, alors que les rectangles colorés en représentent les formes terminales. L'axiome *START* est décomposé en quatre éléments de façade *F*, eux-mêmes décomposés en une fenêtre *F*, un linteau *KS*, et des murs $enorange$. Les fenêtres *F* sont ensuite décomposées plus avant en un cadre $enbleu$ et une vitre *WIN*, etc. La figure 3.20 montre le résultat de l'application récursives de ces règles.

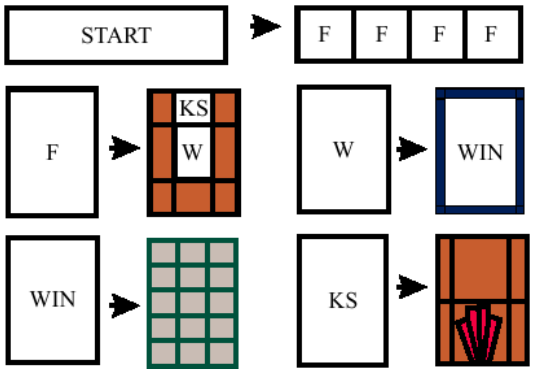


FIG. 3.19 – Exemple de règles d'une grammaire de décomposition [WWSR03]

Les grammaires de décomposition sont par ailleurs accompagnées de grammaires de contrôle qui permettent l'attribution des paramètres des règles de décomposition. En effet, un système de correspondance des attributs pilote le système et sélectionne les règles qui s'appliquent dans les deux grammaires. Les premières s'occupent des formes, les secondes de leurs attributs. Ce qui permet une génération simple, c'est à dire une sélection simple des règles, mais implique une stratégie d'écriture des règles complexes. De plus, la plupart des grammaires de formes existantes ne peuvent pas s'exprimer à l'aide de ce mécanisme qui permet tout de même de modéliser une grande variété de styles de bâtiments et est utilisé dans le cadre de projets d'urbanisme. La figure 3.21 montre

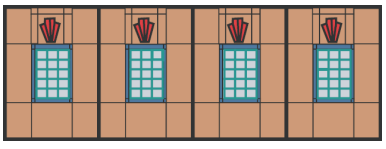


FIG. 3.20 – Résultat de l'application des règles de la figure 3.19

un exemple de scène modélisée à l'aide de ce système.

FIG. 3.21 – Exemple d'environnement modélisé à l'aide de règles de décomposition [WWSR03].

3.4.1.2 Les grammaires de formes architectuales

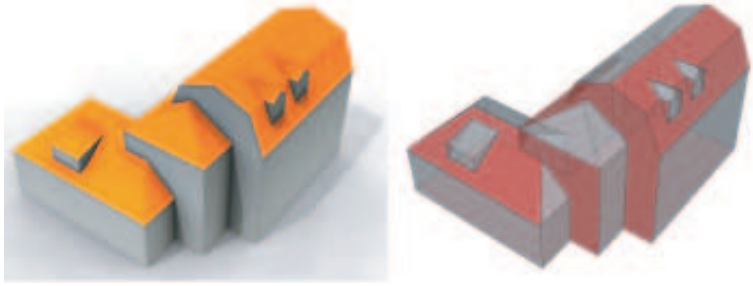


FIG. 3.22 – Modélisation de bâtiments à partir d'un assemblage de volumes simples [MWH$^+$06].

Müller et al. [MWH$^+$06] proposent une adaptation des grammaires de décomposition [WWSR03] appelée *CGA shape* (pour *Computer Graphics Architecture shape*) et que nous avons traduit en grammaire de formes architectuales. Par rapport aux grammaires de décomposition, le langage est beaucoup plus expressif. La première contribution de ces travaux concerne la modélisation des bâtiments à partir d'un assemblage de volumes simples (voir figure 3.22). L'idée elle-même n'est pas nouvelle (voir par exemple [Cor23, Mit90]), mais une stratégie simple consistant à ne conserver que les surfaces et arêtes extérieures est proposée. Ces dernières peuvent alors être marquées par des symboles qui permettent l'application ou non des règles de la grammaire de formes. La seconde contribution concernent la prise en compte, à l'intérieur des règles, de l'occlusion. Il s'agit en fait d'une adaptation de la notion de contexte très courante dans les langages formels (notion de langage contextuel) et en particulier dans les L-systems [PJM94]. Ici, l'occlusion pourra servir à déterminer les formes visibles de la rue par exemple. La troisième contribution importante concerne le *happement* (que nous avons traduit de *snapping*). Ce dernier, illustré par la figure 3.23, permet l'alignement des différentes formes pendant leur décomposition. La figure 3.23 montre l'alignement des fenêtres d'un bâtiment entre ses différentes façades.

Par ailleurs, de nouvelles règles sont proposées pour faciliter l'écriture de grammaires de formes architecturales (les règles *Repeat* et *Subdiv* utilisées sur la figure 3.23

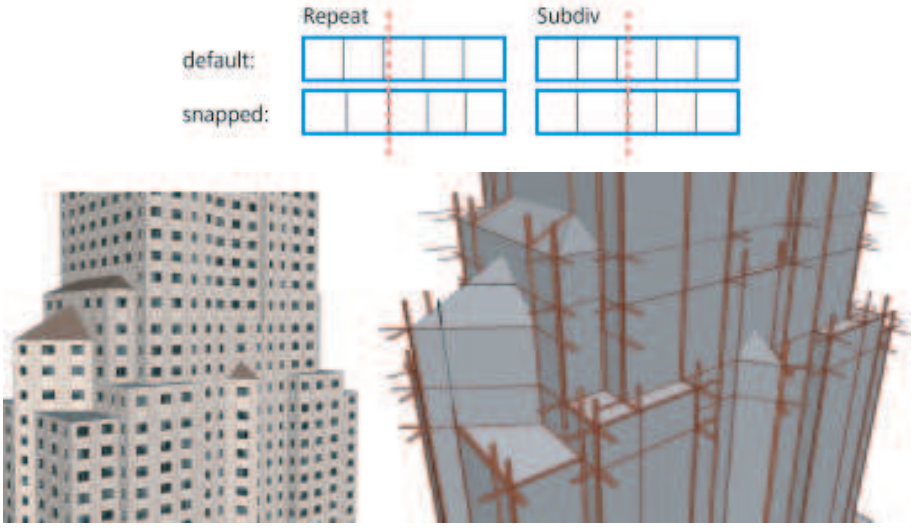


FIG. 3.23 – Utilisation du happement [MWH+06]. En haut : En fonction du type de règle utilisé pour la décomposition, l'effet du happement est différent. Ceci est illustré sur les règles *Repeat* et *Subdiv*, avec et sans happement. En bas : bâtiment dont les fenêtres sont alignées à l'aide du happement.

par exemple. Les résultats produits par cette méthode, illustrés par la figure 3.24, montrent les nouvelles possibilités offertes par ce langage. Néanmoins, on notera que les temps de calcul de chaque modèle de bâtiment sont très longs (une seconde et demi pour chacun des modèles précédents) et que les modèles générés ne permettent pas, à l'heure actuelle, de rendu en temps-réel. Par ailleurs, l'expressivité du langage reste inférieure à celle d'autres travaux ([MPB05] par exemple), et les problèmes posés par l'application de règles de grammaire sur des formes complexes (non convexes par exemple) ne sont pas abordés. Nous reviendrons sur ce dernier points dans les chapitres suivants.

FIG. 3.24 – Résultats de *CGA shape* [MWH+06]. Ici, les mêmes règles sont utilisées sur des volumes différents.

Bekins et al. [BA05] proposent quant à eux l'utilisation de grammaires simples pour définir des *schémas* de façades (facade schema), d'étages (floor schema) et de modèles (model schema) à partir de méthodes de reconstruction. En effet, l'objectif de cette méthode est l'extraction, à partir de photogrammétrie, de textures caractéristiques d'un bâtiment pour générer d'autres bâtiments présentant alors des caractéristiques similaires : des bâtiments du *même style*. La première étape, l'extraction des textures, est similaire à l'approcher de Debevec et al. [DTM96] : il s'agit d'une méthode hybride basée image avec une intervention manuelle de l'utilisateur. Néanmoins, ici, le modèle est subdivisé en surfaces caractéristiques (fenêtres, portes,...). C'est donc seulement la seconde étape, la génération de nouveaux bâtiments, qui est

procédurales. Ces schémas permettent ainsi la définition simple de modèles assez variés. Néanmoins, leur expressivité est limitée à la grammaire initiale.

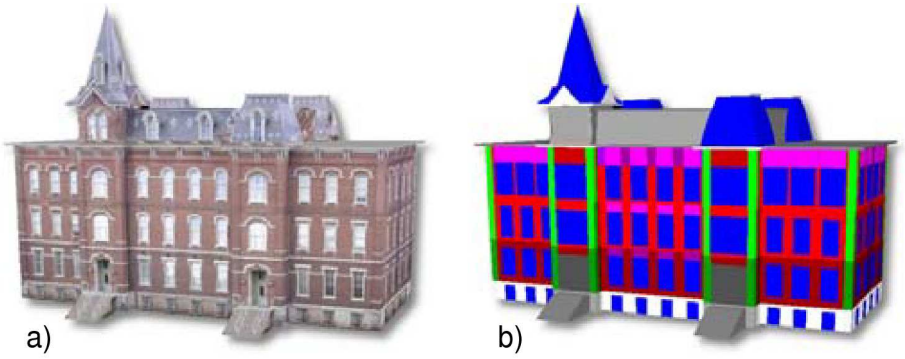


FIG. 3.25 – Illustration de la méthode de Bekins et al. [BA05]. a) Bâtiment reconstruit. b) caractéristiques identifiées sur le modèle.

3.4.1.3 Une autre théorie des formes

Leyton [Ley01a, Ley01b, Ley01c] propose une théorie des formes basée sur le pliage (folding), une autre façon de voir la décomposition. Le principe est basé sur la décomposition du modèle en volumes simples (boules, parallélépipèdes, cylindres) qui seront ensuite décomposés en volumes plus simples. À chaque niveau de la hiérarchie ainsi constituée sont définis des opérateurs de subdivision ou de dépliage (unfolding). Cette théorie présente l'avantage de permettre une étude du volume (cf. 3.26), de la décomposition en étages et en pièces d'un bâtiment en respectant un processus de conception architecturale.

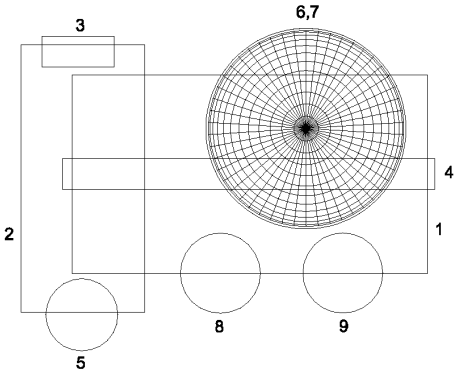


FIG. 3.26 – Étude d'un plan de masse [Ley01a].

3.4.2 Le volume et la modélisation solide procédurale

3.4.2.1 Modélisation solide explicite

Au lieu de traiter l'environnement urbain de façon tout d'abord planaire puis extrudé en volume, il est possible de l'aborder en utilisant les techniques de modélisation solide. Il existe deux principales représentations pour les modèles solides :

- *Constructive Solid Geometry* (CSG) : en géométrie solide constructive, les objets sont construits à partir de primitives simples (parallélépipèdes, cylindres, sphères) qui sont assemblées en hiérarchies par des opérateurs booléens (union, intersection, soustraction).
- *Boundary Representations* (BRep) : les représentations par frontières utilisent des graphes pour modéliser la topologie des solides, ainsi que les opérateurs d'Euler, qui permettent de modifier les objets tout en conservant des propriétés de cohérence (manifold).

FIG. 3.27 – Le langage GML [HF01]. À gauche : une fenêtre gothique générée par application récursive de fonctions génériques. À droite : deux niveaux de détails produits par l'utilisation des surfaces de subdivision sur certains sommets du modèle.

Ces dernières ont été récemment utilisées pour leur compatibilité avec les techniques de subdivision de surfaces de type Catmull/Clark [CC78]. En effet, Havemann [HF01, HF03, HF04] propose le langage de modélisation géométrique procédurale *GML* (pour Generative Modeling Language). Ce langage très expressif est basé sur le langage *Postscript* [Inc99], les *Combined BReps* (CBRep), et les opérateurs d'Euler. Postscript est un langage bas niveau basé pile. Il permet une grande efficacité des modèles et une manipulation minimum de paramètres qui sont en fait directement empilés et dépilés au besoin. Les CBReps étendent les BRep en y ajoutant la définition de propriétés sur les sommets. En effet, on peut ainsi définir des surfaces courbes : des algorithmes de subdivision de surfaces sont utilisés pour subdiviser les polygones dont les sommets sont marqués comme étant arrondis. Les opérations de création et de modification des points, arêtes et surfaces du modèle sont gérées par les opérateurs d'Euler. Berndt [BFH05] propose l'utilisation de tels modèles pour limiter les informations transmises à travers le réseau dans le cadre d'application client-serveur. Day [DAHF03, DAHF04] présente une utilisation de ces modèles pour la modélisation d'environnements urbains virtuels (cf. figure 3.28). Le langage GML propose ainsi d'excellentes performances et une grande expressivité. Néanmoins, c'est un langage de bas niveau et peu approprié à la modélisation par un non spécialiste de l'informatique graphique.

D'autres approches en modélisation solide, comme la modélisation des surfaces implicites ou les techniques volumiques, permettent de modéliser et d'animer des objets dits *mous* (soft) comme des nuages, de la fumée, du feu, qui sont difficilement modélisables à l'aide de techniques traditionnelles [EMP+03].

3.5 Modélisation procédurale du mouvement : un modèle dynamique

Les méthodes de modélisation procédurales permettent ainsi de modéliser le développement d'organismes biologiques ainsi qu'un grand nombre d'objets géométriques par complexification de leur géométrie. De plus, ces modèles peuvent être utilisés pour créer des animations du développement des organismes ou objets étudiés. Néanmoins, dans la plupart des méthodes, l'animation et la modélisation forment deux composantes distinctes d'une application et sont définies en utilisant des formalismes différents. Il ne s'agit pas ici de l'animation du développement d'un arbre par

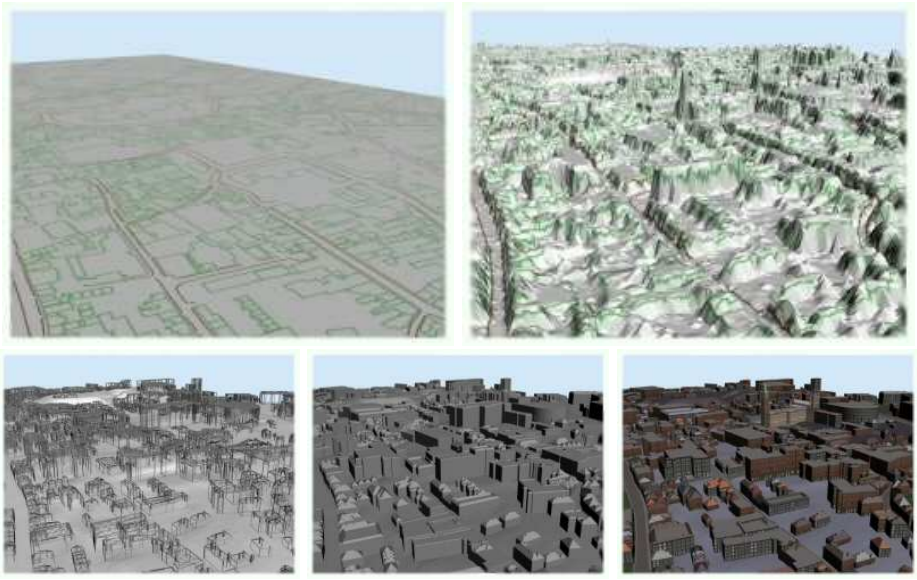


FIG. 3.28 – Reconstruction d'un environnement urbain par l'UMG (Urban Modeling Group) [DAHF04].

exemple, mais du balancement de ses branches et feuilles au gré du vent, de l'animation de son mouvement plutôt que de son développement. Perbet [Per03] propose une approche par complexification combinant animation et modélisation. L'outil de modélisation qu'il présente, *Dynamic Graph* est basé sur des *amplifieurs*. Un amplifieur est une fonction qui ajoute des détails à un objet. Le créateur du modèle écrit en fait des *générateurs d'amplifieurs* qui, comme leur nom le suggère, créent des amplifieurs. Afin d'obtenir de bonnes performances dans le cadre de la navigation temps-réel, ces amplifieurs sont créés à la demande de la visualisation pour générer les détails visibles (cf. 3.2). Dans son exemple de prairie animée, chaque brin d'herbe est généré par un amplifieur lui même généré par un générateur : la prairie. Les générateurs sont fournis sous la forme de programmes C++ contenant les fonctions de construction, d'affichage et d'amplification. Ainsi, pendant la navigation, les amplificateurs sont évalués à chaque pas de temps, permettant ainsi l'animation des brins d'herbes.

Cet outil est très intéressant et utilise de façon efficace le modélisation procédurale dans le cadre de la visualisation temps-réel. Néanmoins, l'utilisation de C++ pour la modélisation d'objets géométriques paraît difficilement utilisable dans la pratique, même sans considérer les arguments de Dijkstra [Dij72] en faveur des langages spécialisés. Par ailleurs, le créateur doit ici gérer à la fois la modélisation géométrique, l'animation et la visibilité en même temps. Il paraît en effet raisonnable de traiter ces problèmes indépendamment. De plus, le modèle ne semble pas adapté à la modélisation du développement, ce qui est dommage compte tenu que le modèle présenté le plus abouti est celui d'un arbre.

3.6 Modélisation de l'évolution : «Chomsky rencontre Darwin» ou les modèles génétiques

La conception assistée par ordinateur s'est intéressée à l'évolution depuis de nombreuses années. Bentley [Ben99] montre les intérêts de cette conception dite évolutionnaire en quatre points :

1. l'évolution est une méthode générale de résolution de problèmes qui s'applique à de nombreuses applications,
2. les algorithmes évolutionnaires ont été utilisés avec succès dans de nombreux domaines de recherche (fouille de données, théorie des jeux, apprentissage, ...),
3. l'évolution partage de nombreuses caractéristiques avec le processus de conception humain,
4. les plus remarquables créations connues de l'homme sont le produit de l'évolution naturelle qui constitue l'inspiration à l'origine des algorithmes évolutionnaires.

L'ingénierie génétique définit les notions de *génotype* ou *génome* et de *phénotype*. Le génotype désigne l'ensemble des informations constituant le code génétique. Du point de vue de l'ingénierie, le génotype est un ensemble d'instructions génériques, c'est à dire un langage formel. Le phénotype, quand à lui, représente l'ensemble des caractères visibles. Il est le produit de l'interaction du génotype et de son milieu, c'est à dire la structure générée par l'interprétation du génotype. Ainsi, à partir d'un code génétique définit par un utilisateur, il est possible de combiner les instructions ou règles existantes pour en obtenir de nouvelles et ainsi produire de nouveaux phénotypes ou modèles.

Les méthodes évolutionnaires ont été utilisées pour la conception d'objets tridimensionnels très divers [Ben99]. Rosenman [Ros97] propose l'utilisation de grammaires simples pour la génération de plans 2D de maisons à un étage. Le mécanisme de réécriture utilisé est basé sur une grille et la réécriture des arêtes des polygones formant le plan. Bentley [Ben99] propose l'utilisation de génotypes formés de hiérarchies de séquences de bits pour la génération d'objets quelconques, de la table à café à l'hôpital. Une approche similaire est proposée par Saleri [Sal03, MBST01] pour la génération automatique de modèles de villes (voir la figure 3.29). En utilisant cette même représentation binaire du génotype, Schnier et Gero [SG96] présentent une alternative à l'écriture manuelle des grammaires. En effet, dans le contexte des grammaires de formes (présentées en 3.4.1), ils montrent qu'il est possible d'apprendre automatiquement une grammaire à partir d'exemples de plans architecturaux. Ils obtiennent ainsi une grammaire représentant le génotype d'un style architectural. De nombreux problèmes restent tout de même à résoudre puisque peu de mots générés présentent des plans valides (on obtient par exemple un porche à l'intérieur de la maison). Jackson [Jac02] propose l'utilisation des L-systems pour une approche coévolutionnaire pour la génération d'architectures. Son modèle co-évolutionnaire présente une façon de faire se co-adapter des espèces qui évoluent séparément. Cette approche, très expérimentale, permet de prendre en compte un contexte.

Marsault et al.[MBST01] utilisent le codage par *IFS* (Iterated Functions Systems) de l'empreinte au sol des bâtiments pour coder une image du plan d'une ville. Ce codage permet alors de regénérer le plan en question. Ils utilisent ensuite une approche génétique pour croiser différents génotypes de villes.

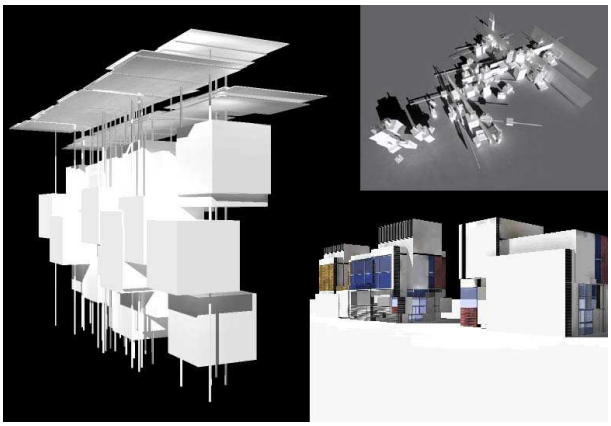


FIG. 3.29 – Un générateur de bâtiments aléatoires d'après Saleri [Sal03].

3.7 Discussion

Nous avons ici présenté de nombreux formalismes très puissants pour modéliser les environnements virtuels. Après avoir présenté un état de l'art de la modélisation procédurale urbaine, on peut identifier :

- trois échelles temporelles : évolution, développement, mouvement,
- trois échelles spatiales urbaines : géographique, urbaine, architecturale.

Ces échelles se retrouvent dans la plupart des modèles présentés. Le tableau 3.30 présente les principales approches abordées dans ce chapitre ainsi que leurs échelles correspondantes. Nous avons par ailleurs ajouté l'échelle biologique. Comme nous le voyons dans cette figure, aucune méthode de modélisation procédurale ne permet, à notre connaissance, de modéliser la ville en prenant en compte toutes ces échelles. Il s'agit bien là de la difficulté de la modélisation des multi-objets, c'est à dire des objets existant dans plusieurs dimensions. Nous n'avons par ailleurs pas trouvé, dans la littérature, d'approche procédurale prenant en compte les particularités historiques, sociales et culturelles dans la modélisation urbaine. Mais revenons sur les principaux éléments des environnements urbains présentés ci-après.

Les réseaux de transport

Nous avons vu deux principales approches dans la modélisation des réseaux de transport urbains, c'est à dire l'échelle urbaine : les L-systems [PM01] et les automates cellulaires [Tor00]. Néanmoins, ni les L-systems ni les automates cellulaires ne semblent adaptés à la modélisation urbaine à tous ses niveaux. En effet, les L-systèmes, s'il peuvent modéliser l'évolution d'un réseau de transport, semblent peu appropriés pour la modélisation architecturale. En effet, les propriétés biologiques des L-systèmes peuvent gêner la modélisation d'objets dont le développement n'est pas continu, comme les bâtiments. Nous verrons dans le chapitre suivant que ce problème n'est pas insurmontable. Les automates cellulaires ne travaillent quant à eux que sur des grilles, ce qui limite beaucoup leur expressivité et la qualité des modèles engendrés.

Méthode	évo	dév	mvt	géo	urb	arch	bio
L-systems [Lin68]	×	×					×
L-systems urbains [PM01]				×	×	×	
Graphtals [MBST01]	×				×	×	
Grammaires de formes [Kni00]						×	
Grammaires de décomposition [WWSR03]						×	
Modélisation générative [HF04]						×	
Modélisation générative urbaine [DAHF04]					×	×	
Dynamic Graph [Per03]		×	×				×
Approche évolutionnaire [Ben99]	×					×	×

FIG. 3.30 – Les méthodes procédurales et leurs échelles associées. Les trois premières colonnes concernent les échelles temporelles déjà mentionnées : *évo* pour évolution, *dév* pour développement, *mvt* pour mouvement. Les quatres colonnes suivantes concernent les échelles spatiales : *géo* pour géographique, *urb* pour urbaine, *arch* pour architecturale, et *bio* pour biologique.

Les bâtiments

En ce qui concerne les bâtiments, c'est à dire l'échelle architecturale, de nombreuses méthodes existent. Néanmoins, les L-systems offrent les plus importantes capacités de généralisation. En effet, seuls les L-systems ont été utilisés avec succès à toutes les échelles spatiales, de la modélisation d'organismes multi-cellulaires aux réseaux urbains.

Dans le chapitre suivant, nous présentons ainsi une extension des L-systems offrant plus de souplesse : les *FL-systems*. Ces systèmes permettent, entre autres, d'étudier les processus de constructions qui sont l'équivalent du développement pour les organismes biologiques. De plus, nous explorons les rapports entre les règles utilisées par les *FL-systems* et la représentation de connaissance. Pour finir, nous essayerons de montrer que les *FL-systems*, grâce à leur généralité et leur indépendance vis à vis des symboles, pourraient répondre aux besoins des différentes échelles spatiales et temporelles composant la ville.

Chapitre 4

FL-systems : un langage de modélisation fonctionnel

Dans ce chapitre, nous proposons les FL-systems, un langage fonctionnel de modélisation géométrique. Comme nous l'avons vu dans le chapitre précédent, les L-systems sont difficiles à utiliser pour la modélisation d'objets non biologiques. En effet, étant basé sur la modélisation du développement, ils peuvent paraître inadaptés à la modélisation d'objets créés par l'homme tels les bâtiments. Wonka et al. [WWSR03] déclarent à ce sujet que les bâtiments ne croissent pas. Sans déclarer pour autant que les bâtiments croissent, nous pensons qu'ils peuvent être modélisés à l'aide de systèmes de croissance. En effet, la motivation des systèmes de croissance comme les L-systems est l'application de règles de réécriture afin de simuler le développement dans le temps des objets étudiés. Si, en effet, les bâtiments ne croissent pas, ils sont bien construits. Ainsi, nous pouvons modéliser ce phénomène de construction à l'aide d'un modèle de développement discret dans le temps, chaque étape de construction correspondant à une étape de développement.

La motivation principale des FL-systems est finalement la souplesse. En effet, les L-systems manquent de souplesse car les objets géométriques générés par homomorphisme sont implicites, c'est à dire que l'utilisateur doit en fait modifier la fonction qui effectue l'homomorphisme pour pouvoir modifier les propriétés de ces objets. De plus, si ce n'est dans le modèle proposé par Prusinkiewicz pour la modélisation de phénomènes continus, le temps est, lui aussi, exprimé de façon implicite, ce qui ne permet pas de contrôler des séquences simplement. D'autre part, la génération d'une liste d'éléments ne peut se faire que par récursivité, ce qui n'est pas toujours naturel ou efficace en termes algorithmiques.

Les FL-systems étendent donc le formalisme proposé par Prusinkiewicz en y ajoutant la notion d'objets génériques comme paramètres, le contrôle de l'application des règles, et surtout l'utilisation de fonctions comme terminaux et comme opérateurs sur les paramètres.

La structure de ce chapitre est la suivante : nous présentons tout d'abord les motivations qui nous ont poussés à développer les FL-systems. Ensuite, nous proposons une définition formelle de ces derniers, ainsi que les différentes extensions que nous avons apportées. La section 4.3 présente des applications des FL-systems dans la pratique : à la modélisation de bâtiments, à la modélisation d'environnements urbains, de processus de construction et de règles de composition. Un cache pour les FL-systems est proposé en section 4.4, suivi par une discussion générale sur ce formalisme.

4.1 Motivations

Les L-systems sont lents

Notre première constatation concerne le processus de réécriture et d'évaluation des L-systems. En effet, de façon classique, un L-system est un système de réécriture de mots. Il est ainsi généralement traité comme une chaîne de caractères. Initialement, la chaîne de caractères contient uniquement l'axiome. Ensuite, à chaque étape de réécriture, les non-terminaux (ou têtes de règles) sont remplacés par leur partie droite où les paramètres formels sont évalués en paramètres effectifs. Le résultat d'un tel processus est donc une chaîne de caractères d'autant plus longue que le L-system est complexe et le nombre d'étapes de dérivation important.

L'évaluation du L-system, c'est à dire l'application de l'homomorphisme, consiste alors à parcourir cette chaîne de caractères obtenus en évaluant les terminaux rencontrés. Cette étapes nous permet d'obtenir une image, un objet 3D ou une séquence d'instructions *OpenGL*.

Exemple 6 *Reprenons à titre d'exemple le L-system présenté en section 3.3.6 :*

ω : F

$p1$: $F \overset{0.33}{\rightarrow} F[+F]F[-F]F$

$p2$: $F \overset{0.33}{\rightarrow} F[+F]F$

$p3$: $F \overset{0.34}{\rightarrow} F[-F]F$

Une dérivation après trois étapes de réécriture de ce L-system est la suivante :
$F[+F]F[+F[-F]F]F[+F]F[-F[+F]F]F[+F]F[-F]F$

Afin d'obtenir un modèle à partir de cette chaîne de caractères, nous pouvons, par exemple, utiliser des primitives OpenGL *simples telles qu'un cylindre suivi d'une translation pour le terminal F, un empilement et un dépilement de la matrice* OpenGL *pour les terminaux $[$ et $]$ respectivement et des rotations pour $+$ et $-$. Pour ce faire, il nous faut donc parcourir la chaîne de caractères et appeler les primitives correspondantes aux terminaux rencontrés. Le résultat d'un tel processus sur cet exemple est illustré figure 4.1.*

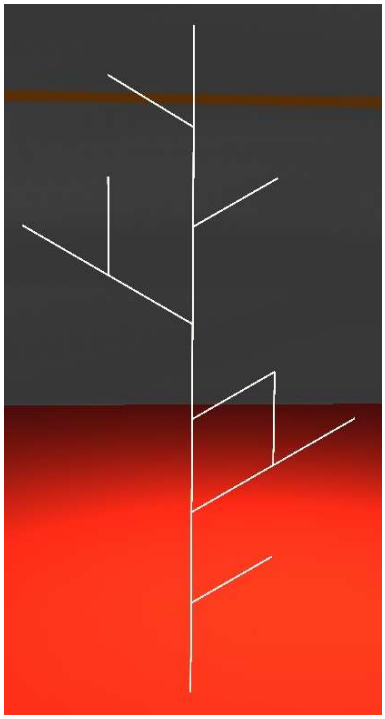


FIG. 4.1 – Modèle généré par l'exemple 6.

La construction puis le parcours de cette chaîne de caractères peuvent être coûteux en temps de calcul, et nous constatons que la seconde étape, l'évaluation, peut être évitée si la construction du modèle 3d ou de la séquence d'instructions est faite

au fur et à mesure de la réécriture. En effet, les terminaux qui produisent les objets ou les instructions pendant l'évaluation ne sont pas réécrits pendant la construction. Ainsi, en évaluant ces terminaux pendant la construction, nous effectuons les deux étapes en une seule fois.

Par ailleurs, une telle évaluation permet d'utiliser les propriétés hiérarchiques des règles de réécriture et permet la construction de boites englobantes hiérarchiques par exemple, qui permettent une plus grande efficacité des modèles engendrés.

La sémantique des règles est perdue

Notre seconde constatation concerne les arbres d'évaluation des L-systems. En effet, une fois la construction de la chaîne de caractères terminée, il ne reste plus d'information concernant la façon dont les terminaux ont été obtenus ou la règle par laquelle ils ont été générés. Ceci ne pose pas de problème concernant l'évaluation, mais empêche toute traçabilité des terminaux. La traçabilité des objets, dans les modèles procéduraux, est une propriété importante car essentielle pour pouvoir corriger un éventuel problème dans le modèle procédural. En effet, savoir par quelles règles ont étés générés les terminaux permet de localiser rapidement les erreurs et de les corriger.

Pour finir, l'utilisation de l'arbre de réécriture permet d'utiliser les méthodes formelles telles que le calcul dynamique de dépendances. Cette technique nous permet de définir un cache pour les FL-systems et d'améliorer les performances globales de la réécriture.

Les paramètres sont insuffisants

Les nombres (réels ou entiers) sont parfois insuffisants pour modéliser des objets complexes. En effet, pour plus de liberté dans les paramètres il est intéressant de disposer d'objets génériques et plus particulièrement de listes, vecteurs, matrices et de chaînes de caractères. Nous avons intégré ces objets aux paramètres des FL-systèmes.

La réécriture parallèle est parfois gênante

En effet, nous avons intégré un deuxième type de réécriture équivalent aux règles de décomposition afin de pouvoir contrôler avec plus de finesse et de précision le processus de réécriture.

Les paramètres implicites sont limitatifs

Traditionnellement, le temps est implicite dans les L-systems. Ce qui peut être limitatif lorsque certains évènements doivent être déclenchés à des moments précis.

4.2 Définition

Définition 10 (FL-system) *Un* FL-system *est un quintuplet ordonné* $G =< V, \Sigma, \omega, P, \pi >$ *où :*

- $V = \{A - Za - z\}\{A - Za - z0 - 9\}^*$ *est l'alphabet du système.*

- $\Sigma = V$ *est l'ensemble des paramètres formels.*
- $\omega \in (V \times \mathcal{P}^*)^+$ *est un mot paramétré non vide dit axiome. Dans l'axiome, tous les paramètres formels sont affectés (ce sont alors les paramètres du mot paramétré).*
- $P \subset (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times (\mathcal{E}(\Sigma) \cup \mathcal{F}(\Sigma))^*)^*$ *est un ensemble fini de productions où :*
  - $(V \times \Sigma^*)$ *est l'ensemble des prédécesseurs de la production.*
  - $(V \times (\mathcal{E}(\Sigma) \cup \mathcal{F}(\Sigma))^*)^*$ *est l'ensemble des successeurs de la production.*
  - $\mathcal{C}(\Sigma)$ *est une* expression logique *utilisée comme condition d'application de la production (parfois appelée garde).*
  - $\mathcal{E}(\Sigma)$ *est une* expression arithmétique *sur les paramètres formels de la production.*
  - $\mathcal{P} = \mathcal{R} \cup \mathcal{O} \cup \mathcal{S}$ *est l'ensemble des paramètres effectifs de la production.*
  - $\mathcal{R}$ *est l'ensemble des nombres réels.*
  - $\mathcal{O}$ *est l'ensemble des objets génériques incluant les vecteurs à* $n \times m$ *dimensions quelconques, et le pointeur.*
  - $\mathcal{S}$ *est l'ensemble des chaines de caractères.*
  - $\mathcal{F}(\Sigma)$ *est l'ensemble des fonctions à paramètres dans* Σ *et à valeur dans* $\mathcal{P}$.
- $\pi : P \rightarrow (0,1)$ *est la* distribution de probabilité *qui fait correspondre à l'ensemble des productions des* probabilités de production.

Ainsi, l'alphabet V*, l'axiome* ω *et la distribution de probabilité* π *sont définis comme dans un 0L-system stochastique (définition 8). Néanmoins, l'ensemble des productions* P *est redéfini.*

Un FL-system possède principalement les mêmes propriétés qu'un L-system non contextuel (ou 0L-system) telles que les règles paramétrées, les conditions et les règles stochastiques. Un exemple de chacun de ces types de règles est donné ci-dessous :

règle paramétrée	$A(x,p)$	$\rightarrow$	$B(x,p)D(p)$
règle conditionelle	$B(x,p) : x = 0$	$\rightarrow$	$D(p)$
règle stochastique	$D(p)$	$\stackrel{p}{\rightarrow}$	E
	$D(p)$	$\stackrel{1-p}{\rightarrow}$	F

Par ailleurs, les FL-systems offrent de nouveaux types de règles :

nouvel alphabet	$Tr(tdeb, tfin, dt) : tdeb < tfin$	$\rightarrow$	$Tr2(tdebut + dt, tfin)$
chaines de caractères	$Branche(r,l)$	$\rightarrow$	$Cylindre(r,l,''bois.png'')$
fonctions-paramètres	$Liste(l,i) : i > 0$	$\rightarrow$	$Liste(conc(l,i), i-1)$
fonctions-terminaux	$RotX(matr, angle)$	$\rightarrow$	$rotate(mat, 1, 0, 0, angle)$

Le nouvel alphabet permet de définir un plus grand nombre de règles et de leur attribuer des noms plus significatifs : il est plus simple de se rappeler le rôle de la règle $Branche$ que de la règle B. L'utilisation de chaines de caractères comme paramètres permet, par exemple, d'inclure les noms des textures appliquées aux objets géométriques dans la grammaire. L'ajout de fonctions comme paramètres augmente l'expressivité du langage avec, par exemple, les listes et leurs fonctions de modification associées. Finalement, les fonctions utilisées comme terminaux du langage permettent la génération d'objets géométriques et l'appel de fonctions pendant le processus de réécriture. Nous reviendrons plus en détail sur cette propriété.

À l'aide de ces nouvelles règles, il est ainsi possible de définir une plus grande diversité de règles et d'utiliser de nouveaux paramètres. Par exemple, il devient possible d'utiliser le même FL-system avec différentes textures en changeant juste un paramètre de l'axiome, comme l'illustre la figure 4.2.

FIG. 4.2 – Utilisation de textures dans l'axiome d'un FL-system.

4.2.1 Approche fonctionnelle

Contrairement aux L-systems, les FL-systems possèdent une distinction entre terminaux et non-terminaux. En effet, la transformation entre chaîne de caractères et modèle géométrique se fait par homomorphisme dans le cadre des L-systems. Ainsi, c'est seulement lorsque la réécriture est terminée qu'un non-terminal devient, s'il possède un homomorphisme, un terminal. Dans l'exemple 6, le non-terminal F devient terminal lorsque la réécriture est terminée et un homomorphisme lui attribue la géométrie d'un cylindre. Contrairement au module F, le module $+$ peut être considéré comme terminal puisqu'aucune règle ne lui est affectée. Une fois encore, la distinction entre terminal et non-terminal est artificielle dans les L-systems.

Au contraire, dans les FL-systems, à l'instar des grammaires de Chomsky, il existe une distinction forte entre terminaux et non-terminaux : les non-terminaux sont l'objet de réécriture alors que les terminaux non. De plus, dans le contexte des FL-systems, des fonctions sont assignées aux terminaux. Ainsi, à chaque fois qu'un terminal est rencontré, la fonction qui lui est assignée est exécutée. Ainsi, en observant un ordre de réécriture correct, les terminaux rencontrés successivement exécutent une série d'instructions créant le modèle 3D attendu.

Il existe plusieurs cas de figure en fonction du type d'instructions qu'exécutent les terminaux. Chacun de ces cas de figure implique une stratégie de réécriture différente qui dépend aussi, par ailleurs, du type de contexte traité par les modules du FL-system. Prenons un exemple : nous définissons les terminaux d'un FL-system comme étant des appels à des primitives *OpenGL*. Nous appellerons ce type de FL-system particulier un *GLFL-system*. Dans un *GLFL-system*, il est primordial que les primitives soient invoquées dans l'ordre dans lequel les terminaux se trouvent à la fin

de la réécriture. Il s'agit ainsi d'un parcours en largeur des feuilles de l'arbre d'appel. Le moyen le plus simple d'effectuer un tel parcours est d'utiliser une réécriture récursive à gauche, traditionnelle dans les grammaires de Chomsky. Une telle réécriture est simple à mettre en place : chaque réécriture d'un module est faite de façon récursive et les fonctions associées aux terminaux sont appelées au fur et à mesure qu'elles sont rencontrées. Il se pose alors une question d'expressivité : obtient-on le même langage en utilisant cette technique plutôt qu'en utilisant la réécriture parallèle de Lindenmayer ? La réponse est oui sous deux conditions :

1. Il s'agit d'un FL-system non-contextuel (0FL-system). En effet, les contextes diffèrent en fonction de l'ordre de réécriture.
2. Une attention est portée à l'utilisation d'un paramètre remplaçant la profondeur de réécriture utilisée dans les L-systems. Autrement dit, la profondeur de réécriture, ou l'âge du système, est explicite et utilisée avec précaution.

Exemple 7 (Structure ramifiée modifiée) *Pour illustrer la réécriture d'un GLFL-system, reprenons l'exemple 6 présenté en section 4.1. Nous allons ainsi introduire dans ce L-system deux paramètres : le temps écoulé depuis le début de la réécriture td, qui correspond ici au nombre d'étapes de réécriture effectuées, et le temps final tf, c'est à dire le nombre d'étapes de réécriture total.*

ω	$: F(0,1)$		
$p1$	$: F(td,tf)\ : td < tf$	$\rightarrow$	$F1(td+1,tf)$
$p2$	$: F(td,tf)$	$\rightarrow$	$Cylindre(1,0.1)$
$p3$	$: F1(td,tf)$	$\overset{0.33}{\rightarrow}$	$F(td,tf)\ [+F(td,tf)]\ F(td,tf)\ [-F(td,tf)]\ F(td,tf)$
$p4$	$: F1(td,tf)$	$\overset{0.33}{\rightarrow}$	$F(td,tf)\ [+F(td,tf)]\ F(td,tf)$
$p5$	$: F1(td,tf)$	$\overset{0.34}{\rightarrow}$	$F(td,tf)\ [-F(td,tf)]\ F(td,tf)$

Nous avons par ailleurs ajouté les règles $p1$ et $p2$ afin de transformer, lorsque la réécriture est terminée ($td = tf$) les modules F en terminaux $Cylindre$ et de séparer cette transformation de la dérivation conditionnelle effectuée par les règles $p3$, $p4$ et $p5$. Les fonctions affectées ici aux terminaux $+$, $-$, $[$, $]$ et $Cylindre$ sont particulièrement proches de l'interprétation *tortue* proposée par Prusinkiewicz [PLH+90] (voir section 3.3.4) :

$+$	$glRotate(45,1,0,0)$
$-$	$glRotate(-45,1,0,0)$
$[$	$glPushMatrix()$
$]$	$glPopMatrix()$
$Cylindre(l,r)$	$gluCylinder(gluNewQuadric(),r,r,l,10,1)$ $glTranslatef(0.0,0.0,l)$

La primitive $gluCylinder$ crée un cône tronqué de hauteur l autour de l'axe z, en particulier ici un cylindre puisque les deux rayons (celui à la base en $z = 0$ et celui en $z = l$) sont égaux à r. Les deux derniers paramètres (10 et 1, fixés de façon arbitraire) désignent le nombre de subdivisions respectivement autour et le long de l'axe z. À la fonction $Cylindre$ est ajoutée la primitive $glTranslatef$ qui place le repère à la tête du cylindre ainsi créé. La figure 4.3 illustre l'exécution de cette primitive. Les primitives $+$ et $-$ sont associées à la primitive $glRotate$ pour effectuer une rotation

autour de l'axe x (nous aurions pu tout aussi bien choisir l'axe y). Les terminaux [et] empilent et dépilent respectivement la matrice de transformation utilisée par *OpenGL* pour modéliser la position et l'orientation du repère ou système de coordonnées local.

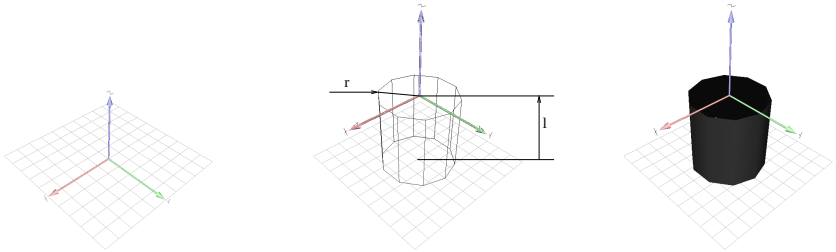


FIG. 4.3 – Primitive $Cylindre$, de gauche à droite : le repère avant l'invocation de la primitive, le cylindre crée en fil de fer et avec remplissage des polygones après invocation de la primitive. On notera la nouvelle position du repère à la tête du cylindre.

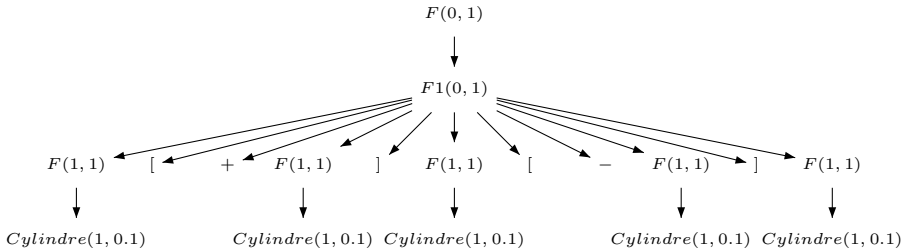


FIG. 4.4 – La dérivation du FL-system l'exemple 7.

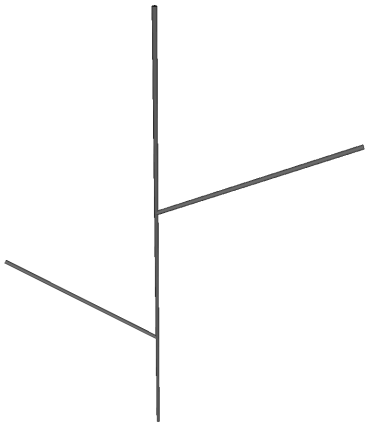


FIG. 4.5 – Modèle généré par la dérivation de l'exemple 7.

La dérivation de l'exemple 7 est illustrée par la figure 4.4. Nous observons sur cet exemple que les terminaux '[', '+', ']' et '−' sont rencontrés avant les terminaux $Cylindre$. Ainsi, pour que l'ordre des opérations soit correct, il convient ici d'utiliser

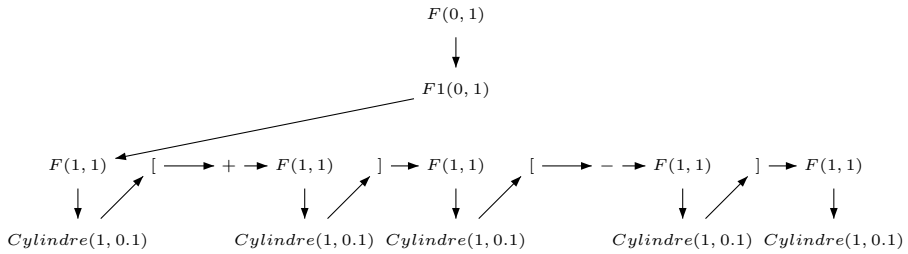


FIG. 4.6 – La dérivation récursive du FL-system l'exemple 7.

une réécriture récursive à gauche, illustrée par la figure 4.6. Le résultat d'une telle réécriture sur l'exemple est illustré par la figure 4.5.

Dans cette section, nous avons présenté l'approche fonctionnelle des FL-systems et montré, en utilisant l´exemple des GLFL-systems, que l'expressivité du langage obtenu par cette approche en utilisant une réécriture récursive est équivalente à celle d'un 0L-system. Par ailleurs, elle permet d'effectuer en une seule étape la réécriture du FL-system et l'application de l'homomorphisme associé. Néanmoins, il est à la charge de l'utilisateur de gérer les paramètres temporels rendus explicites dans les FL-systems alors qu'ils sont le plus souvent implicites dans le cadre des L-systems. Ceci n'est pas sans rappeler l'extension comportant les règles de décomposition proposée par Prusinkiewicz [Pru03] (voir section 3.3.3).

4.2.2 Contrôle de la réécriture

La réécriture récursive n'est pas adaptée à toutes les utilisations possibles des FL-systems. Par exemple, elle ne permet pas la prise en compte du contexte. En effet, la récursivité à gauche implique qu'à la réécriture de tout non-terminal, son contexte à gauche ne contienne que des terminaux. Ainsi, la prise en compte des non-terminaux qui figurent à sa gauche dans le cadre d'une réécriture parallèle est impossible. Il en est de même pour la réécriture récursive à droite et le contexte droit. Pour permettre un éventuel compromis, de nouvelles méthodes de réécriture doivent être mises en place. Nous proposons dans cette section une méthode de contrôle de la réécriture utilisant des opérateurs de synchronisation. Cette approche a été utilisée pour la définition d'une extension permettant la création de graphes de scène *VRML97* [MCB97][1].

4.2.2.1 Une extension pour le *VRML97*

Dans le cadre des L-systems ou des GLFL-systems, la structure du modèle 3D engendré correspond à la structure du système qui le génère. En effet, ce sont les terminaux qui créent, par homomorphisme ou par appel de fonctions, les éléments composant le modèle. Néanmoins, la structure d'un graphe de scène *VRML* n'est

[1] Nous parlerons par la suite de *VRML* pour désigner *VRML97*.

pas adaptée à un tel mécanisme. Les transformations (translations, rotations ou changement d'échelle), par exemple, ne peuvent être effectuées, en *VRML*, que pour un sous-graphe entier. Ainsi, une simple série de terminaux telle que $F + F + F$ (en considérant qu'à F est associé un cylindre et à $+$ une rotation) nécessite la définition de la hiérarchie illustrée par la figure 4.7 en *VRML*.

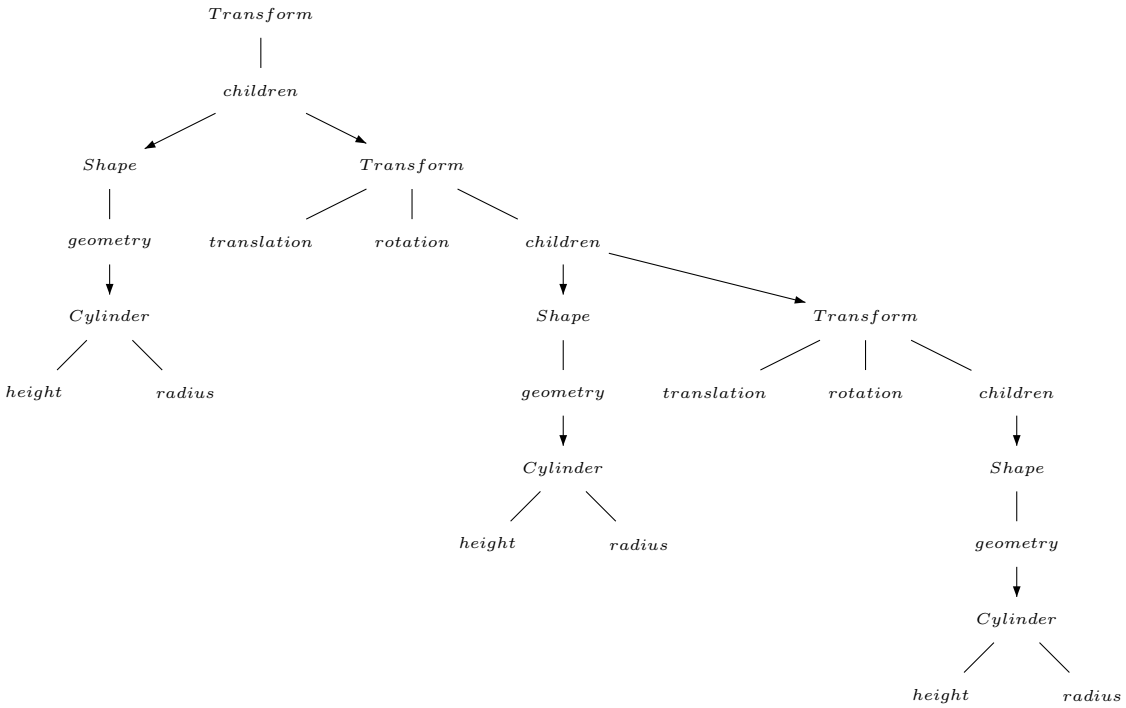


FIG. 4.7 – Graphe VRML équivalent au L-system $F + F + F$.

Comme on peut le constater sur la figure, le graphe de scène *VRML* correspondant au L-system $F + F + F$ est relativement complexe par rapport au système qui le génère et présente surtout une structure très différente. Contrairement au cas des GLFL-systems ou des L-systems, la génération de graphes de scène *VRML* implique ainsi une séparation importante entre la structure du générateur (le système) et la structure des données générée (le graphe de scène). Pour manipuler les données ainsi créées, nous introduisons dans le langage trois fonctions spécifiques à VRML :

- *defVrmlNode(type,nom)* : cette fonction, qui crée des nœuds *VRML*, possède deux paramètres qui correspondent respectivement au type de nœud *VRML* créé (Transform, Shape, Cylinder dans l'exemple précédent) et à son nom, qui est optionnel. Cette fonction est utilisée comme paramètre de règles. A l'exécution, elle crée le nœud et retourne comme résultat un pointeur vers le nœud créé.
- *setVrmlField(nœud, nomChamp, valeur)* : cette fonction affecte la valeur du champ *nomChamp* à un nœud *VRML*. Elle possède trois paramètres étant le nœud *VRML*, le nom du champ affecté et sa valeur. Cette fonction est un terminal du langage dont le premier paramètre *nœud* est le résultat d'un appel à *defVrmlNode*.

– *useVrmlNode(nœud)* : cette fonction permet l'instanciation d'un nœud défini à l'aide de la fonction *defVrmlNode*. Elle est utilisée comme paramètre de règle.

Ces trois fonctions simples permettent la création d'un graphe de scène *VRML* complet. Néanmoins, il faut noter qu'une fois encore la réécriture parallèle n'est pas adaptée : en effet, il est préférable, afin de conserver un graphe de scène bien formé, de s'assurer de l'affectation de tous les champs d'un nœud avant d'utiliser ce nœud et de l'insérer dans le graphe de scène. Aussi, en utilisant la réécriture parallèle, s'assurer d'une telle propriété peut être délicat. Nous proposons dans la section suivante une extension des L-systems permettant de contrôler de façon très fine et très simple la synchronisation de certaines réécritures.

4.2.2.2 Un opérateur de synchronisation

Afin de contrôler la réécriture des règles, nous introduisons le symbole *'!'* comme opérateur de synchronisation. Cet opérateur, dans le cadre d'une réécriture parallèle, retarde la réécriture du module (terminal ou non-terminal) qui le suit jusqu'à la réécriture complète des modules qui le précèdent. On pourrait ainsi appeler cet opérateur un opérateur d'attente. Pour illustrer le fonctionnement de cet opérateur, nous proposons l'exemple 8.

Exemple 8 (Utilisation de l' opérateur de synchronisation) *Dans cet exemple, nous reprenons le L-system illustré par la figure 4.7 en utilisant les terminaux* VRML *introduits précédemment. La règle* $Three$ *construit un cylindre à l'aide de la règle* $Cylinder$*. Le nœud* VRML *correspondant au cylindre créé est contenu dans le nœud shape. La règle* $Three$ *affecte ensuite les valeurs des champs* $rotation$*,* $translation$ *et* $children$ *des nœuds* $t1$ *et* $t2$ *conformément à la figure 4.7. Nous en avons profité pour illustrer l'utilisation explicite de l'instanciation : en effet, alors que le nœud* $t2$ *utilise le nœud shape, les nœuds* $t1$ *et* $root$ *en utilisent une instance à l'aide de la fonction* $useVrmlNode$*. On notera que le module* $Cylinder$ *contient les trois terminaux* $setVrmlField$*, qui affectent les champs des nœuds* cyl *et* $shape$*, sans utiliser d'opérateur de synchronisation. En effet, les terminaux situés au même niveau de la réécriture sont traités de façon séquentielle. Par contre, les terminaux* $setVrmlField$ *présents dans la règle* $Three$ *sont précédés du symbole de synchronisation. Celui-ci est utile puisqu'il faut que les champs des nœuds définis par le module* $Cylinder$ *soient affectés pour être utilisés par les nœuds* $t1$*,* $t2$ *et* $root$*. Ainsi, cet exemple produit bien le graphe de scène* VRML *illustré figure 4.7.*

$$
\begin{array}{lll}
\omega & : Three(1, 0.1, 45, defVrmlNode(Transform, root), defVrmlNode(Shape, myCylinder), & \\
 & \quad defVrmlNode(Transform, t1), defVrmlNode(Transform, t2)) & \\
p1 & : Three(l, r, a, root, shape, t1, t2) \quad \rightarrow & Cylinder(l, r, shape, defVrmlNode(Cylinder)) \\
 & & !setVrmlField(t2, children, (shape)) \\
 & & !setVrmlField(t2, rotation, (a, 1, 0, 0)) \\
 & & !setVrmlField(t2, translation, (0, 0, l)) \\
 & & !setVrmlField(t1, children, (useVrmlNode(shape), t2)) \\
 & & !setVrmlField(t1, rotation, (a, 1, 0, 0)) \\
 & & !setVrmlField(t1, translation, (0, 0, l)) \\
 & & !setVrmlField(root, children, (useVrmlNode(shape), t1)) \\
p2 & : Cylinder(l, r, shape, cyl) \quad \rightarrow & setVrmlField(cyl, height, l) setVrmlField(cyl, radius, r) \\
 & & setVrmlField(shape, geometry, cyl)
\end{array}
$$

4.2.2.3 Intégration à une plate-forme de rendu

Grâce à l'opérateur de synchronisation et à l'extension pour *VRML*, les FL-systems ont été intégrés à la plate-forme *Magellan* [Mar01, MPB03, Mar04, MPB05]. Cette plate-forme, que nous ne détaillerons pas ici, est une application de rendu temps-réel possédant une architecture de type client/serveur. Nous avons ainsi implémenté une extension à cette application permettant d'utiliser les FL-systems. Chaque FL-system est inclus dans un fichier *VRML*. À cette fin, nous avons défini un type de nœud *VRML* appelé *MagellanGrammar* possédant deux champs : l'axiome *axiom* du FL-system et l'ensemble des règles *grammar*. La distinction entre ces deux champs nous permet notamment d'utiliser le mécanisme de prototypes de *VRML*. En effet, en définissant un prototype contenant seulement les règles et ayant comme paramètre l'axiome du FL-system, ce dernier peut être instancié de nombreuses fois avec des axiomes différents. Les FL-systems chargés par Magellan sont lus et traités par un processus parallèle au processus de rendu. Ainsi, chaque ajout de nœud dans le graphe de scène par le processus de réécriture est pris en compte au fur et à mesure par le rendu (voir figure 4.8).

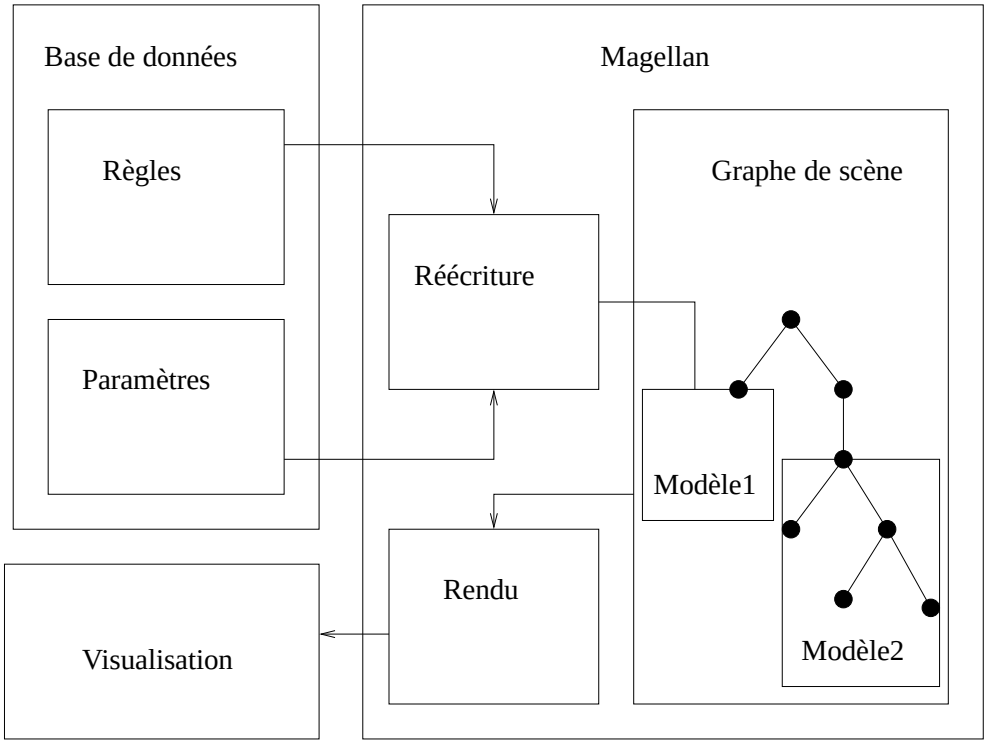


FIG. 4.8 – Processus de réécriture à la volée. Les FL-systems sont analysés et réécrits par un processus en parallèle au processus de rendu. Au fur et à mesure de la réécriture, les nœuds *VRML* sont ajoutés au graphe de scène.

4.2.2.4 Conclusion

Dans la section 4.2.2, nous avons présenté une extension des FL-systems pour la génération de graphes de scène *VRML* au fil de la réécriture ainsi qu'un opérateur de synchronisation permettant l'utilisation de cette extension pour le rendu temps-réel.

Ce travail, effectué en collaboration avec Jean-Eudes Marvie [MPB03, MPB05], a été appliqué à la modélisation de bâtiments et à la modélisation d'environnements urbains. Dans le cadre de ces travaux, la plate-forme de rendu *Magellan* est issue des travaux de Jean-Eudes Marvie. Notre contribution concerne en effet l'extension de cette application à la génération de géométrie à l'aide de *FL-systems*. Ces applications sont présentées dans la section suivante.

4.3 Applications

4.3.1 Modélisation de bâtiments

Comme nous l'avons montré précédemment, les méthodes procédurales permettent la description de familles de modèles géométriques. Nous présentons ici la possibilité d'utiliser les FL-systems dans le cadre de la modélisation de styles architecturaux. Nous allons illustrer cet exemple à l'aide de deux bâtiments situés place des lices à Rennes (figures 4.11 et 4.12). Pour modéliser le premier bâtiment, nous avons tout d'abord décomposé la façade en éléments architecturaux. Ces éléments sont des objets paramétrés par leur largeur et leur hauteur.
L'axiome du FL-system étudié possède comme paramètres :

- l'empreinte au sol du bâtiment (sous la forme d'une liste de points 3d),
- le nombre d'étages du bâtiment,
- la hauteur du rez-de-chaussée,
- la hauteur des étages suivants,
- la largeur des modules horizontaux.

La hauteur totale du bâtiment est calculée facilement à partir du nombre d'étages et de leurs hauteurs. La largeur de chaque façade est déterminée par la distance entre deux points consécutifs de l'empreinte au sol. À partir de la largeur d'une façade $largeur$ et de la largeur des modules horizontaux $largeur_{module}$, le nombre de modules n est calculé par le calcul suivant : $n = \lfloor (largeur - 2 * calage_{min}) / largeur_{module} \rfloor$, où $calage_{min}$ désigne la largeur minimale du calage. La largeur du calage $calage$ est alors déterminée par : $calage = (largeur - n * largeur_{module})/2$. Ces deux calculs, intégrés dans le FL-system, définissent la trame de la façade illustrée par la figure 4.9.

La figure 4.10, quant à elle, donne un apperçu du FL-system ainsi définit. Ne ne présentons ici qu'une partie des règles, en particulier celles se référant aux calculs présentés auparavant, en laissant de côté celles concernant la génération et l'initialisation des nœuds *VRML*. La règle *facade* produit la géométrie d'une façade du modèle. Ses paramètres contiennent toutes les dimensions des éléments géométriques générés. Les calages (*wedging*) font ainsi en sorte, grâce à leur élasticité, que la façade tienne dans les dimensions imparties (la largeur en particulier). La règle *facade* transmet ses paramètres à la règle *layout* : *m*, le nombre d'étages, *n*, le nombre de modules horizontaux (formés d'une fenêtre et d'un trumeau), *w*, la largeur du module, *H*, la hauteur du rez-de-chaussée, *a*, la lageur d'une fenêtre, *d*, la demi-largeur des pilastres, *h* la hauteurs des étages supérieurs, et *s*, la largeur du *calage*. La règle *layout* utilise deux règles pour séparer le traitement du rez-de-chaussée et des autres étages. Le rez-de-chaussée est généré par la règle *firstFloor* et décomposé en *n arcades* et deux calages (*wedging*), un de chaque côté de la façade. La règle *nextFloors* traite tous les autres étages, placant les calages (*wedging*), pilastres (*pilaster*), fenêtres (*window*) et

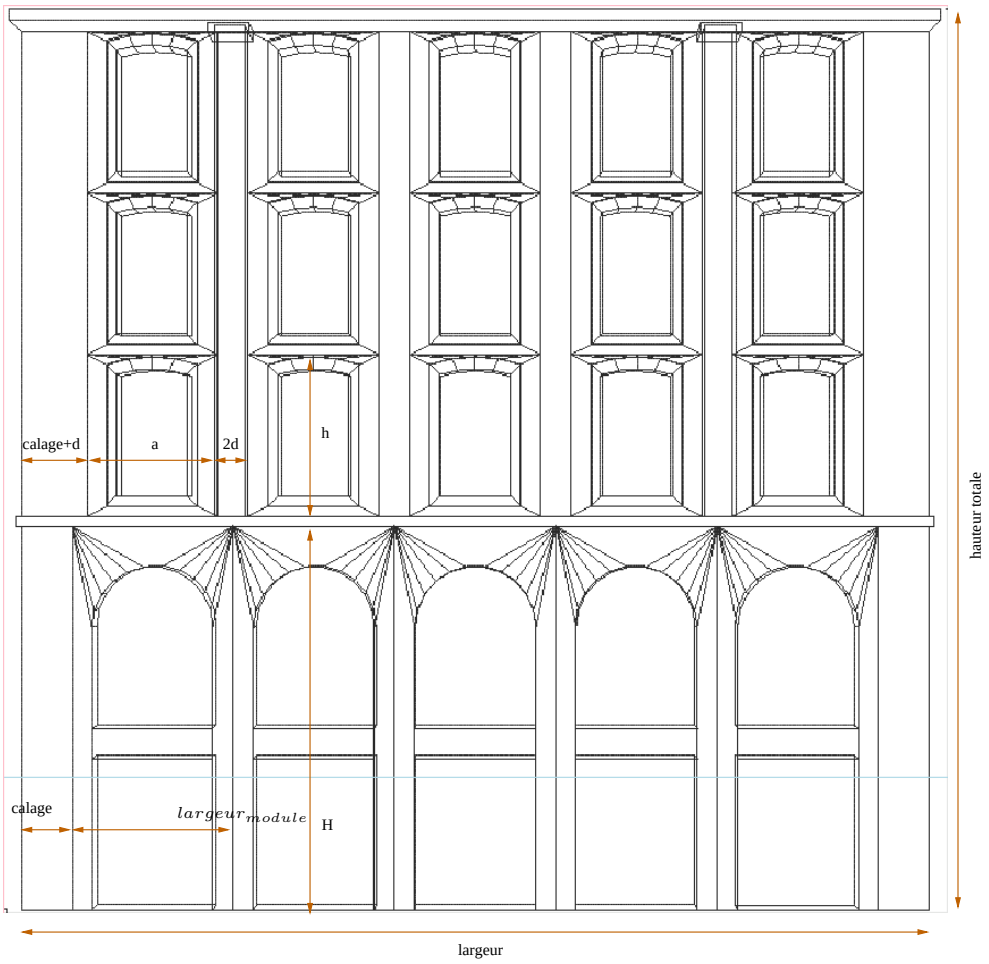


FIG. 4.9 – Composition de la façade en fonction de ses paramètres.

trumeaux (*pier*).

Nous avons extrait manuellement à partir d'une photographie les textures des fenêtres et arcades du bâtiment étudié et évalué les paramètres d'entrée du FL-system pour obtenir la reconstruction du bâtiment présentée figure 4.11. Le même processus a été appliqué à moindre coût au bâtiment illustré figure 4.12. En effet, ces deux bâtiments, situés sur la même place, présentent des structures similaires et le même FL-system a pu être utilisé pour les deux à l'exception des paramètres de l'axiome et du jeu de textures utilisé. D'autres exemples de bâtiments engendrés par ce FL-system sont illustrés figure 4.13. Pour illustrer la réutilisabilité des règles des FL-systems, la figure 4.13 illustre un modèle de la tour de Pise. Dans ce modèle, les règles utilisées pour modéliser les arcades vues précédemment ont été réutilisées. La principale modification apportée, outre le jeu de textures, concerne les règles de composition des modules qui sont placés circulairement au lieu d'être placés autour d'un polygone.

L'utilisation de FL-systems, basés sur un processus de croissance, pour la modélisation de bâtiments peut être mise en question. On pourrait, par exemple, douter de l'existence d'un phénomène de croissance dans le processus de construction d'un

---

facade$(width, m, w, H, a, d, h, smin...) =$
layout$(m, n = floor((width - 2 * smin)/w), w, H, a, d, h, (width - n * w)/2, ...)$
$...;$
layout$(m, n, w, H, a, d, h, s, ...) =$
firstFloor$(n, w, H, s, ...)$
nextFloors$(m - 1, n, w, H, a, d, h, s, ...)$
$...;$

firstFloors$(n, w, H, s, ...) =$
wedging$(0, H, s, ...)$
for $(i = 0; i < n; i++)$ *arcade*$(i, n, w, H, ...)$
wedging$(s + n * w, H, s, ...)$
$...;$

nextFloors$(m, n, w, H, a, d, h, s, ...) =$
wedging$(0, H, s + d, m * h, ...)$
pilaster$(s + d + a, H, 2 * d, m * h, ...)$
for $(i = 0; i < m; i++)$
for $(j = 0; j < n; j++)$
window$(s + d + j * w, H + i * h, ...)$
for $(j = 2; j < n - 1; j++)$
pier$(s + j * w - d, H, 2 * d, m * h, ...)$
pilaster$(s + (n - 1) * w - d, H, 2 * d, m * h, ...)$
wedging$(s + n * w - d, H, s + d, m * h, ...)$
$...;$

...

FIG. 4.10 – Extrait du FL-system utilisé pour générer les figures 4.11, 4.12 et 4.13. Nous avons ici utilisé respectivement les noms $width$, s et w pour désigner les variables $largeur$, $calage$ et $largeur_{module}$ utilisées dans la figure 4.9.

FIG. 4.11 – Reconstruction d'un bâtiment situé place des lices à Rennes.

bâtiment. Il ne s'agit, en effet, pas de croissance à proprement parler. Néanmoins, il est intéressant de constater la similitude entre la structure des arbres de réécriture utilisés pour la modélisation des bâtiments présentés ici et le processus de croissance

FIG. 4.12 – Reconstruction du second bâtiment.

des plantes. Pour illustrer cette similitude, nous avons utilisé cette même structure en remplaçant les portes, arcades et fenêtres par des branches et des feuilles. Le résultat, présenté figure 4.14 nous conforte dans les perspective d'une telle approche. Par ailleurs, nous pensons que le processus de construction des bâtiments peut être simulé par un processus de croissance, ce qui sera l'objet de futurs travaux.

4.3.2 Intégration à la modélisation urbaine

Nous pouvons désormais utiliser les FL-systems définis pour la modélisation de bâtiments dans un contexte urbain. À cette fin, nous avons implémenté un logiciel, *CityZen*, qui génère des modèles urbains. Les modèles créés sont destinés à être visualisés en utilisant la plate-forme *Magellan*. Ainsi, *CityZen* génère des fichiers *VRML* en utilisant le mécanisme de prototype défini dans *VRML*.

CityZen procède en plusieurs étapes comme suit :

1. Création d'une grille sous la forme d'un graphe régulier non orienté ;
2. Suppression aléatoire de nœuds du graphe et des arcs associés ;
3. Modification aléatoire des positions des nœuds du graphe ;
4. Calcul des boucles du graphe, c'est à dire des îlots ;
5. Calcul des carrefours et des routes ;
6. Calcul des empreintes au sol des bâtiments, choix des parcs ;
7. Attribution des hauteurs et styles des bâtiments ;
8. Génération des cellules de navigation ;
9. Génération des fichiers *VRML*.

4.3.2.1 Création du réseau routier

Les trois premières étapes du processus sont illustrées figure 4.15 avec un graphe contenant 6×6 nœuds. Ces étapes permettent la création d'un réseau routier simple et régulier. De plus, la structure du réseau facilite le calcul des cycles les plus courts du graphe, qui définissent les îlots urbains, c'est à dire les zones entourées de routes. Ceci constitue la quatrième étape.

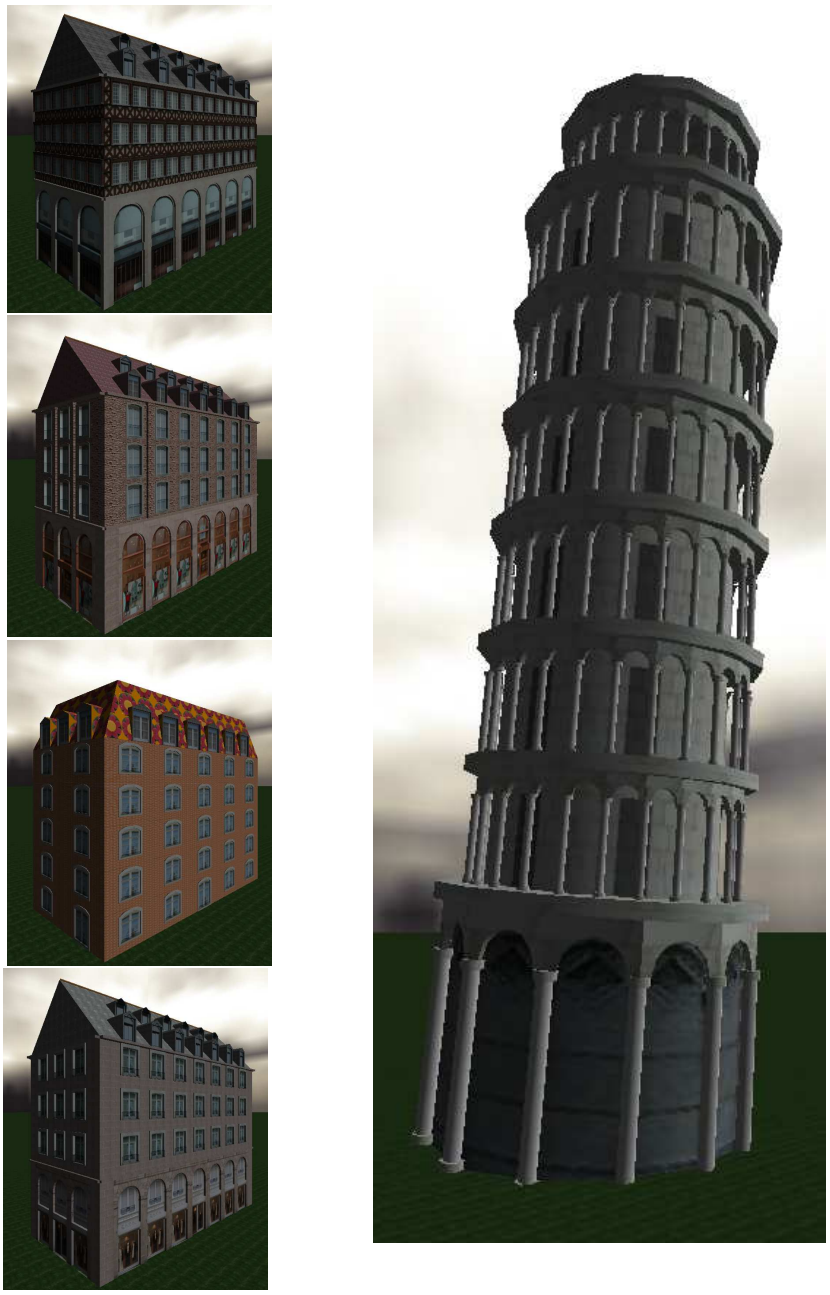


FIG. 4.13 – À gauche : D'autres exemples de modèles engendrés par le FL-system. À droite : Un modèle de la tour de Pise réutilisant les arcades des modèles précédents.

4.3.2.2 Calcul des contours des carrefours, routes et îlots

Une fois les îlots trouvés, la cinquième étape calcule les contours des carrefours et routes, et incidemment ceux des îlots. Ce calcul, illustré par la figure 4.16 consiste à calculer l'intersection de deux rayons. Pour le croisement C, deux rayons $R1$ et $R2$ sont construits parallèlement aux routes qui leur sont associées et translatés des largeurs $W1$ et $W2$ correspondant aux demi-largeurs des rues ajoutées aux largeurs des trottoirs. Ceci nous donne le point $P1$ d'intersection des rayons $R1$ et $R2$. Cette même opération est répétée pour tous les couples de routes successives autour du carrefour C pour définir le polygone constituant le contour du carrefour C. Le ré-

FIG. 4.14 – Le bâtiment illustré figure 4.11 et une plante grimpante utilisant un FL-system similaire. Cet exemple suggère l'intérêt d'une approche de la modélisation de bâtiments à l'aide d'un phénomène de croissance.

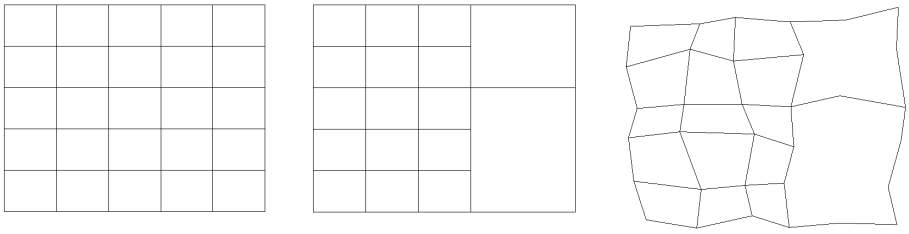


FIG. 4.15 – Réseau routier généré par *CityZen*. De gauche à droite : grille initiale, suppression aléatoire de nœuds, modification des positions des nœuds.

seau étant un graphe, les contours des carrefours nous suffisent pour déterminer les contours des routes et des ilots.

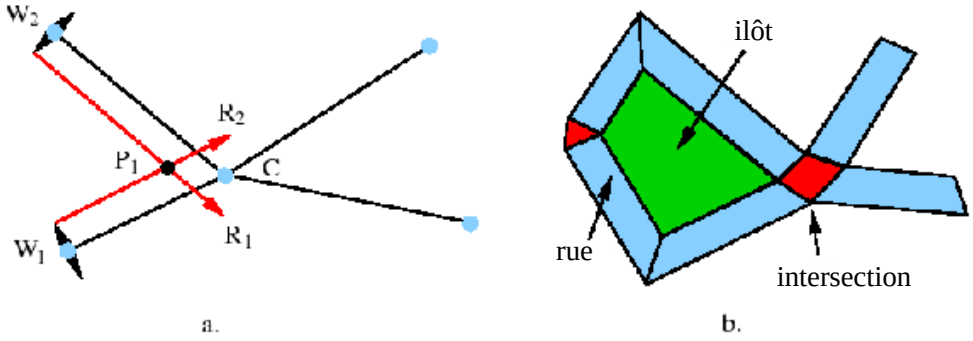


FIG. 4.16 – Calcul des contours des carrefours dans *CityZen*.

4.3.2.3 Choix des parcs et calcul des empreintes

Lors de la sixième étape, une partie des îlots sont choisis aléatoirement pour devenir des parcs. Les autres îlots sont décomposés en empreintes de bâtiments et en cours intérieures à l'aide de l'algorithme illustré par la figure 4.17. Cet algorithme consiste à calculer tout d'abord les contours des cours intérieures. Pour ce faire, nous utilisons le même algorithme utilisé pour l'étape précédente en décalant les rayons de la profondeur des bâtiments. Ensuite, les points obtenus sont projetés sur les deux contours qui leur sont associés pour obtenir les empreintes des bâtiments d'angle. Il ne reste plus ensuite que des rectangles qui sont décomposés en utilisant la largeur moyenne des bâtiments. À la fin de cette étape, nous obtenons une liste d'empreintes au sol de bâtiments marqués comme étant des bâtiments d'angle ou non. Le résultat de cette étape sur le graphe présenté précédemment est illustré en figure 4.18.

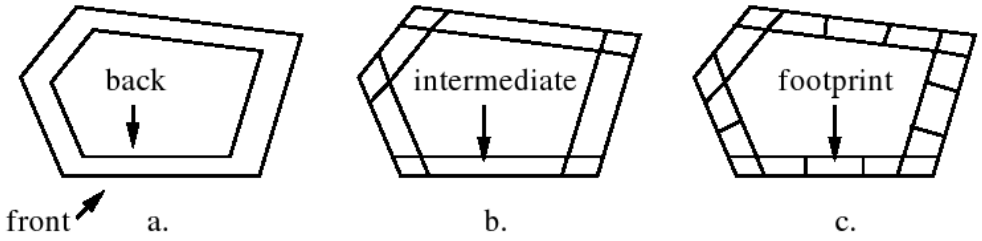


FIG. 4.17 – Calcul des empreintes au sol des bâtiments à partir des ilôts dans *CityZen*.

4.3.2.4 Attribution des paramètres et génération du modèle

En ce qui concerne les bâtiments, les paramètres de la septième étape sont une liste de FL-systems auxquels sont associés une probabilité, et un intervalle pour les hauteurs des bâtiments. Pour les parcs, nous retrouvons une liste de FL-systems et leurs probabilités, ainsi que l'âge moyen des plantes générées. Cette étape n'est pas détaillée puisqu'elle consiste en une répartition aléatoire. Une fois cette étape terminée commence la création des fichiers *VRML* associés. Pour chaque bâtiment, chaque parc et chaque route, un fichier est généré. Un exemple de tels fichiers est donné figure 4.19. Cet exemple décrit un bâtiment comprenant 2 étages. Le paramètre `footprint` définit l'empreinte au sol tandis que le paramètre `adjacentHeights` décrit les hauteurs des bâtiments adjacents. Ce dernier paramètre est calculé grâce à la connexité des empreintes au sol calculée à l'étape précédente. Les hauteurs nulles désignent l'absence de bâtiment pour une certaine façade du bâtiment. Ce paramètre est utilisé par le FL-system pour ne produire que les parties visibles des façades. Ceci permet d'éviter des erreurs comme une fenêtre au niveau du toit d'un autre bâtiment et permet aussi de réduire le nombre de polygones engendrés.

4.3.2.5 Cellules et visualisation

Une fois cette base générée, elle peut être visualisée en utilisant *Magellan*. Les résultats de cette génération sont illustrés figure 4.20. Néanmoins, la visualisation de tels modèles, contenant plusieurs millions de polygones, reste difficile, même à l'aide

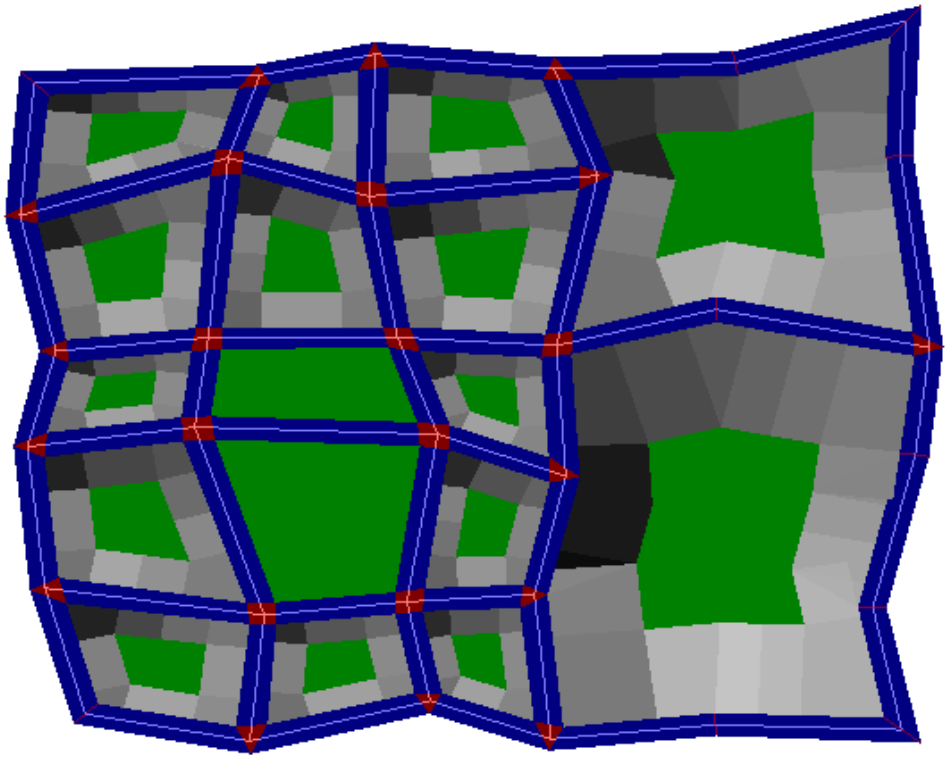


FIG. 4.18 – Un modèle généré par *CityZen*, incluant les empreintes des carrefours, routes, bâtiments et parcs.

des cartes graphiques de dernière génération. C'est pourquoi des algorithmes sont utilisés pour prendre en compte la visibilité dans les environnements complexes. En effet, nous avons découpé les parties navigables de la scène produite en cellules. Les carrefours sont ainsi des cellules à part entière. Par contre, nous avons découpé les routes en trois cellules pour améliorer les performances de navigation. Pour chaque cellule, un ensemble d'objets potentiellement visibles (*Potentially Visible Set*) est calculé et permet des performances bien meilleures dans le rendu des scènes générées. La plate-forme ainsi que le calcul de la visibilité ont été développés par Jean-Eudes Marvie [MPB03, Mar04, MPB05], alors que les FL-systems et la création du réseau urbain (avec *CiyZen*) constituent ma contribution. Grâce à l'utilisation de la visibilité et des FL-systems, il est ainsi possible de naviguer en temps réel dans des environnements virtuels urbains complexes. Pour les performances de la navigation, on pourra se référer au rapport de recherche concerné [MPB03].

4.3.3 Modélisation du processus de construction

Nous n'avons, pour l'instant, pas illustré la capacité des *FL-systems* à modéliser le processus de construction. En effet, nous avons avancé plus tôt que la motivation temporelle des *L-systems*, à vocation de simuler la croissance des organismes biologiques, pouvait aider à modéliser les processus de construction. Nous présentons ici un exemple pour illustrer cette possibilité à travers la construction d'un mur de briques.

```
#VRML V2.0 utf8

# Extern PROTO invocation

EXTERNPROTO Building01 [
   field MFVec3f footprint
   field SFInt32 floors
   field MFInt32 adjacentHeights
]
‘‘Library/building01.wrl’’

# model instanciation

Building01{
   footprint [26.55 0.20 665.99 30.24 0.20 653.32
              9.42 0.20 647.27 16.55 0.20 669.24]
   floors 2
   adjacentHeights [0,6,0,8]
}
```

FIG. 4.19 – Exemple de fichier *VRML* généré pour la description d’un bâtiment. Le fichier mentionné (Library/building01.wrl), contient le prototype *VRML* et le FL-system associé au bâtiment *Building01*. Il s’agit ici d’un FL-system similaire à celui présenté en figure 4.10.

FIG. 4.20 – Images d’un modèle engendré par *CityZen*.

Exemple 9 (Construction d’un mur de briques)

ω $\mathbf{Mur(0, 220, 1, 10, 100, 0.2, 0.04, 0.08, 0.005, 0.008)}$

$p1$ $\mathbf{Mur(tdeb, tfin, tb, m, n, l, h, p, jh, jv)}$: $\mathbf{tdeb < tfin \& n > 0}$
→
$[Rang(tdeb, tfin, tb, m, n, l, h, p, jh, jv)]$ Translate(0, 0, h)
$[Joint(tdeb + m * tb, tfin, m * (l + jh) - jh, jv, p)]$ Translate(0, 0, jv)
$Mur(tdeb + tb * (m + n\%2), tfin, tb, m, n - 1, l, h, p, jh, jv)$

$p2$ $\mathbf{Rang(tdeb, tfin, tb, m, n, l, h, p, jh, jv)}$: $\mathbf{n\%2 == 0}$
→
$Rangee(tdeb, tfin, tb, m, n, l, h, p, jh, jv)$

$p3$ $\mathbf{Rang(tdeb, tfin, tb, m, n, l, h, p, jh, jv)}$: $\mathbf{n\%2 != 0}$
→
$Brique(tdeb, tfin, (l - jh)/2, h, p, rand(0.01))$ $Joint(tdeb + tb, tfin, jh, h, p)$
$Rangee(tdeb + tb, tfin, tb, m - 1, n, l, h, p, jh, jv)$ $Joint(tdeb + m * tb, tfin, jh, h, p)$
$Brique(tdeb + m * tb, tfin, (l - jh)/2, h, p, rand(0.01))$

$p4$ $\mathbf{Rangee(tdeb, tfin, tb, m, n, l, h, p, jh, jv)}$: $\mathbf{tdeb < tfin \& m > 1}$
→
$Brique(tdeb, tfin, l, h, p, rand(0.01))$ $Joint(tdeb + tb, tfin, jh, h, p)$
$Rangee(tdeb + tb, tfin, tb, m - 1, n, l, h, p, jh, jv)$

$p5$ $\mathbf{Rangee(tdeb, tfin, tb, m, n, l, h, p, jh, jv)}$: $\mathbf{tdeb < tfin \& m == 1}$
→
$Brique(tdeb, tfin, l, h, p, rand(0.01))$

*Dans cet exemple, nous avons souligné les prédécesseurs des règles en gras et le seul terminal (*Translate*) a été laissé en police normale.*
Ce FL-system *simule les étapes de construction d'un mur de briques. Les paramètres de l'axiome sont les suivants :*

- $tdeb$ *: instant initial de la construction*
- $tfin$ *: instant final de la construction*
- tb *: temps de pose d'une brique*
- m *: nombre de briques composant une rangée*
- n *: nombre de rangées de briques composant le mur*
- l *: longueur d'une brique*
- h *: hauteur d'une brique*
- p *: profondeur d'une brique*
- jh *: largeur du joint horizontal (entre deux briques consécutives)*
- jv *: hauteur du joint vertical (entre deux rangées de briques)*

Ainsi, l'axiome de l'exemple, illustré figure 4.21, simule la construction d'un mur de 10 briques de large et 100 de haut. Dans la figure, seules certaines étapes clefs de la construction sont illustrées, allant jusqu'à l'étape 220 à laquelle la construction n'est pas terminée. La règle $p1$ *engage la construction d'une rangée de brique à condition que le temps ne soit écoulé ou le nombre de rangées atteint. Les règles* $p2$ *et* $p3$ *traitent la distinction entre les rangées de rang paire, ne contenant que des briques pleines, et les rangées de rang impaire, contenant une demi-brique à chaque extrémité. Les règles* $p4$ *et* $p5$ *posent les briques et le mortier entre elles à l'exception de la dernière brique. La fonction* $rand(x)$ *est utilisée pour produire un nombre aléatoire compris entre* 0 *et* x*. Nous avons, pour des raisons évidentes de simplification, fait l'amalgame entre le mortier, tenant les briques entre elles, et le joint appliqué sur le mortier. Nous avons par ailleurs laissé les règles de construction des briques et joints de côté. Nous préciserons tout de même que le dernier paramètre de la règle* $Brique$ *est utilisé pour introduire du jeu dans l'alignement des briques dans le plan du mur, ceci afin d'obtenir un résultat plus naturel.*

On notera que la construction d'une rangée est ici commencée à l'instant estimé où la rangée précédente est terminée (règle $p1$). Ce choix est discutable et ce

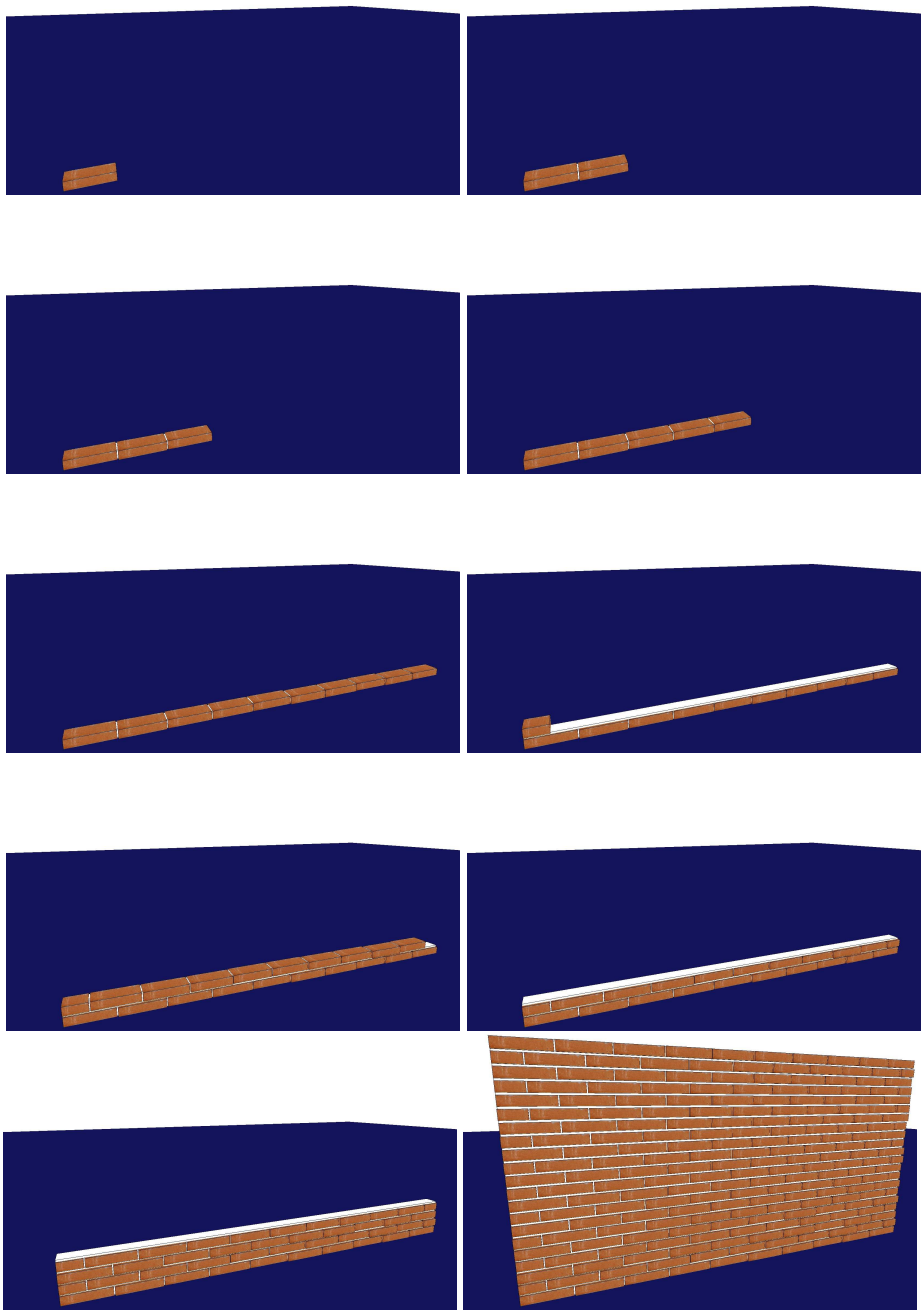


FIG. 4.21 – Construction d'un mur de briques à l'aide du *FL-system* 9. Les étapes illustrées correspondent aux résultat du *FL-system* avec comme paramètre *tfin* 1,2,3,5,10,11,20,21,42 et 220.

mécanisme peut être déclenché de diverses façons, notamment par l'utilisation de signaux [Pru03]. Néanmoins, nous ne présentons ici que le mécanisme le plus simple.

Cet exemple est insuffisant pour pouvoir affirmer que les *FL-systems* permettent de modéliser des processus de construction complexes. Néanmoins, nous pensons

qu'il montre le potentiel d'une telle utilisation. En effet, grâce à l'utilisation de signaux pour déclencher les étapes dépendant de la terminaison d'autres étapes, ainsi que l'utilisation du contexte 3D, nous pensons qu'il est possible de simuler une grande variété de processus de construction. De tels processus seront l'objet de travaux futurs.

4.3.4 Modélisation d'objets architecturaux

Les règles de réécriture des *FL-systems* sont similaires aux règles de production d'un système expert. Ainsi, la réécriture d'un *FL-system* peut être comparée au chainage avant d'un tel système, c'est à dire à une déduction à partir de faits. Dans notre contexte, les faits de départ sont les paramètres du *FL-system*, les faits déduits sont les terminaux, c'est à dire des objets géométriques. Cette analogie met en valeur les corrélations entre *FL-systems* et systèmes experts. En effet, il est possible, à l'aide de *FL-systems*, de définir des bases de connaissances procédurales similaires aux systèmes experts tels que ceux présentés en section 2.4. Nous illustrerons cette propriété par un exemple de colonne de l'ordre Toscan.

Exemple 10 (Définition d'une colonne Toscane)

$p1$	$: Colonne(r)$	$\rightarrow$	$Base(r)\ Fut(r)\ Chapiteau(r)$
$p2$	$: Base(r)$	$\rightarrow$	$Plinthe(r)\ Tore(r)\ Ceinture(r)$
$p2$	$: Chapiteau(r)$	$\rightarrow$	$CeintureChap(r)\ Astragale(r)\ Colarin(r)\ Listel(r)\ Ove(r)\ Abaque(r)$

Ce FL-system *décompose une colonne en éléments architecturaux simples. La règle* Colonne *se décompose en une* Base, *un* Fût *et un* Chapiteau. *La règle* Base *décompose alors la base en une* Plinthe, *un* Tore *et une* Ceinture, *représentés respectivement par un parallélépipède, un tore et un cylindre. De la même façon, la règle* Chapiteau *décompose le chapiteau en objets géométriques simples.*
On notera qu'un seul paramètre r, *le rayon du fût à sa base, est nécessaire pour ce* FL-system, *car, selon les descriptions de Palladio [Pal97] et Vitruve [VP95], une colonne de l'ordre* Toscan *peut être dimensionnée à l'aide de son module seul. En effet, selon la tradition classique, chaque partie d'une colonne toscane est proportionelle à son module. Le module ici, est le diamètre du nu du fût à sa base. De même, la colonne a pour hauteur sept fois le module. Néanmoins, malgré les précisions apportées par Palladio, ces proportions laissent place à l'interprétation et nous prions le lecteur érudit de nous pardonner les erreurs introduites ici. Ainsi, les proportions utilisées pour la base ainsi que la colonne finale sont illustrées dans la figure 4.22. Nous avons ici utilisé, à la place du module, la moitié de celui-ci, c'est à dire les rayon du fût à sa base noté* r. *Tous les terminaux utilisés ici, à l'exeption de la plinthe et de l'abaque, utilisent le terminal* $Lathe(r1, r2, r3, r4, h)$. *Ce terminal crée une surface de révolution dont le profil est illustré dans la figure 4.22.*

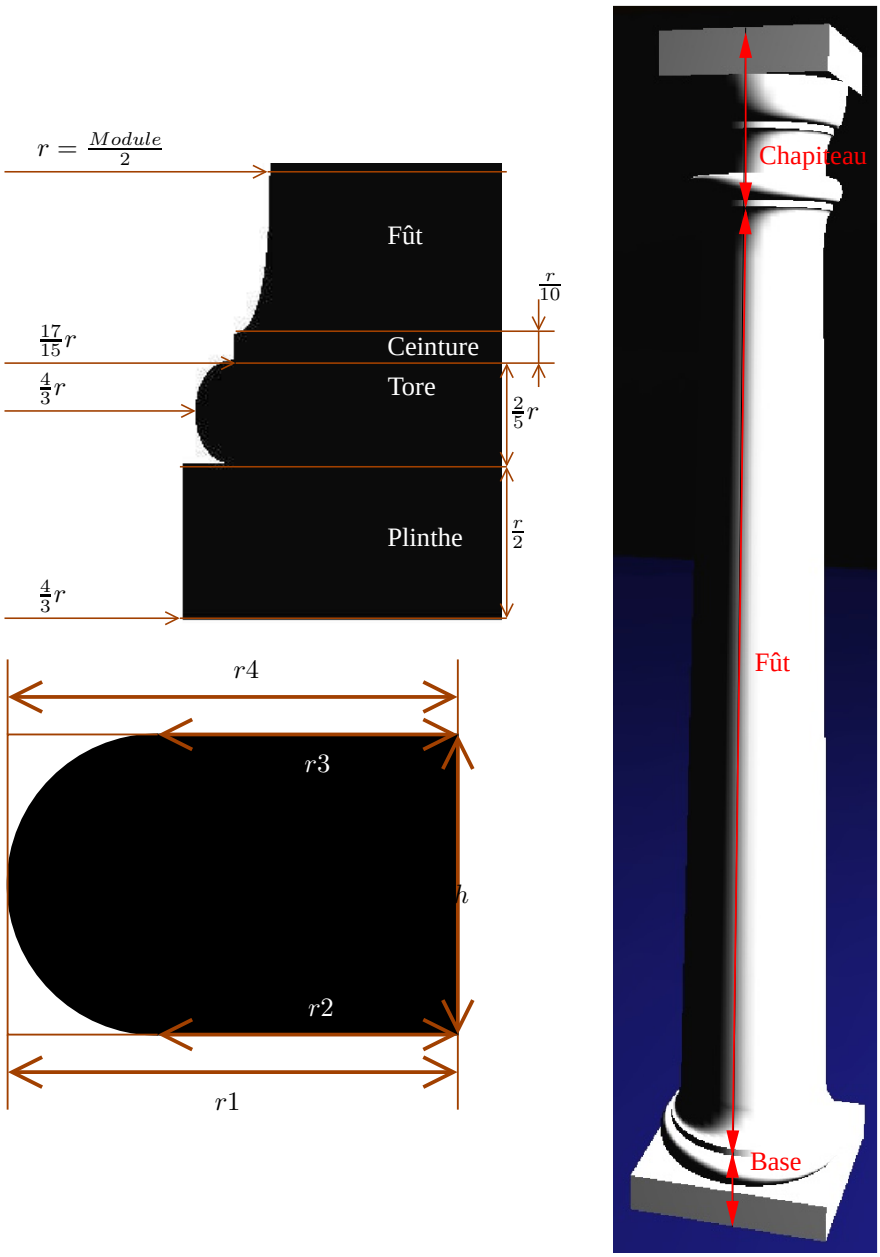


FIG. 4.22 – Définition d'une colonne *Toscane*. En haut, à gauche : proportions utilisées pour la base. En bas à gauche : le terminal $Lathe$ et ses 5 paramètres. À droite : la colonne entière.

Une telle utilisation des *FL-systems* soulève une nouvelle question : si la réécriture des *FL-systems* peut être assimilée au chainage avant d'un système expert, qu'en est-t-il du chainage arrière ? En effet, si l'on est capable, à partir de règles de productions représentant des connaissances, de produire des objets géométriques, est-on capable de renverser le processus et de retrouver les faits initiaux, c'est à dire les paramètres du *FL-system* ? Cette question reste ouverte à l'heure actuelle, mais nous pensons qu'il est possible, dans une certaine mesure, à partir des terminaux produits et du *FL-system*, de trouver un ensemble de paramètres correspondant. Une application de ce problème, par exemple, permettrait de déterminer les paramètres d'un *FL-system* correspondant à des données volumétriques obtenues sur un arbre. C'est

à dire qu'il serait possible, à partir de données volumétriques reconstruites grâce à un ensemble de photos d'un arbre particulier, de trouver un *FL-system* générant précisément cet arbre. Ce problème est similaire à l'analyse de documents à l'aide de grammaires [CCL04]. Par ailleurs, question plus intéressante encore, est-il possible de trouver un *FL-system* générant un résultat donné ? Cette question est bien plus complexe puisqu'il s'agit, dans un premier temps, de déterminer les motifs d'une scène 3D ainsi que leur répétition. Dans un deuxième temps, il faut alors construire un *FL-system* décrivant cette scène.

Au delà d'un débat sur de telles applications futures, la capacité des *FL-systems* à représenter des connaissances (au sens large) est en question. L'exemple 10 décrit en particulier une hiérarchie d'abstraction d'une colonne de l'ordre Toscan. Il s'agit plus précisément d'une généralisation de la colonne Toscane, c'est à dire de la représentation d'une famille d'objets plutôt que d'un objet particulier. De plus, il s'agit aussi d'une hiérarchie d'agrégation : chaque partie de la colonne est décomposé en sous-parties jusqu'aux éléments terminaux représentant ici des éléments de construction. Nous pensons ainsi que les *FL-systems* proposent de nombreuses qualités pour la description d'objets géométriques.

4.3.5 Conclusion

Dans la section 4.3, nous avons présenté une application des *FL-system* pour la modélisation de bâtiments. Nous avons par ailleurs intégré ce travail à la génération d'environnements urbains. Nous avons par ailleurs montré les capacités des *FL-system* dans le cadre de la modélisation de processus de construction et de règles architecturales. De nombreuses études complémentaires restent à effectuer, néanmoins, les résultats obtenus ont déjà montré l'intérêt d'une telle approche. En particulier, les FL-systems présentent des caractéristiques intéressantes en termes de compression de représentation d'objets géométriques présentant une importante auto-similarité. Par ailleurs, comparés aux L-systems, les performances en temps de réécriture sont, à notre connaissance, bien meilleures. En effet, la durée de la réécriture d'un bâtiment dans l'application urbaine est de quelques millisecondes alors que l'édition des liens nécessaire à l'utilisation du langage L+C [KP03] prend à elle seule de l'ordre de la seconde[2]. Néanmoins, pour que cette représentation soit utilisable en tant que représentation géométrique à part entière, nous pensons qu'il est nécessaire d'améliorer ces performances. Dans la section suivante, nous proposons d'utiliser les propriétés formelles des FL-systems afin d'économiser des réécritures inutiles par la mise en place d'un cache dit *intelligent*.

4.4 Utilisation d'un cache intelligent

Les *FL-systems* permettent de modéliser toutes sortes d'objets géométriques et, de préférence, des objets qui présentent une structure fortement redondante, tels des plantes ou des bâtiments. En effet, basés sur une description grammaticale des objets, les *FL-systems* permettent d'utiliser à bon escient la redondance de l'environne-

[2]Le langage L+C, destiné à améliorer les performances des L-systems, nécessite en effet la transformation des L-systems en programmes C, leur compilation en une librairie dynamique et enfin l'édition des liens pour l'utilisation des systèmes ainsi générés.

ment modélisé. Cette répétition permet des descriptions très courtes et génériques de classes d'objets. Une classe d'objets est caractérisée par un ensemble de paramètres. Une instance d'objet est alors obtenue par la définition d'un ensemble de valeurs associées à ces paramètres. Il est alors possible de mettre en place un système de gestion de cache utilisant un axiome et ses paramètres comme clefs et le résultat de la réécriture comme valeur. À chaque fois que le système de réécriture rencontre un axiome et des paramètres dont la valeur a déjà été calculée, on peut alors éviter de recalculer la réécriture.

Néanmoins, les valeurs données en paramètre à l'axiome d'un *FL-system* n'ont pas nécessairement d'influence sur toutes les parties de l'objet généré. Il est en effet courant qu'une même partie d'un objet se retrouve à l'identique dans un autre objet de la même classe, malgré des différences importantes dans les valeurs de leurs paramètres (on pourra ici penser aux feuilles de deux arbres de la même espèce). Dans cette section, nous proposons une grammaire et une sémantique pour les règles de réécriture des *FL-systems* permettant de mettre en place une analyse dynamique des réécritures et d'améliorer ainsi les performance du cache en détectant de tels cas. Ce travail a été réalisé en collaboration avec Gurvan Le Guernic et plus de détails sur celui-ci sont disponibles dans [LGP05]. Ces travaux sont le fruit d'une collaboration très étroite et il m'est impossible de dissocier la contribution de Gurvan Le Guernic de la mienne.

4.4.1 Le paradigme d'instanciation d'objets géométriques

Dans le cadre de la modélisation d'environnements naturels, Deussen et al. [DHL+98] proposent l'*instanciation approximative* comme une extension du paradigme d'instanciation géométrique de Hart [Har92, EMP+03]. Les objets modélisés procéduralement sont projetés dans l'espace des paramètres de l'axiome et ensuite regroupés par *clusters*. Pour chaque *cluster*, un objet est choisi pour représenter alors une instance approximative de tous les objets qui en font partie, réduisant ainsi le nombre d'objets que contient la scène.

Les *FL-systems*, quant à eux, permettent l'utilisation de nœuds *VRML*, et donc l'utilisation du paradigme d'instanciation géométrique. Nous proposons ici un mécanisme d'instanciation exacte qui est automatique, implicite, et transparent pour l'utilisateur. Ce mécanisme est basé sur le calcul dynamique des dépendances dans les règles de réécriture (cf. section 4.4.4). Mais il faut tout d'abord définir la grammaire sur laquelle est basée le calcul des dépendance.

4.4.2 La grammaire utilisée

Comme le montre la figure 4.23, les règles de réécriture des *FL-systems* sont composées d'une partie gauche ($Axiome$), d'une condition ($Cond$), et d'une partie droite ($Terme$). Nous considérerons ici que la stratégie de réécriture est déterministe et récursive à gauche, contrairement aux L-systems dont la réécriture est parallèle (voir section 4.2). Cette restriction a pour but de rendre déterministe la réécriture d'un axiome. De plus, pour contrôler la profondeur de réécriture, nous utiliserons un paramètre donné à l'axiome. Ainsi, à partir d'un axiome donné, une réécriture engendrera un mot composé de terminaux. La grammaire présentée ici reconnait un sous-ensemble des *FL-systems* présentés en section 4.2. L'étude d'un sous-ensemble

se justifie par la simplification du problème et la nature préparatoire des travaux présentés ici.

$Val = \mathbb{R}^+$

$$
\begin{array}{rcl}
Var & ::= & [A-Za-z][A-Za-z0-9]^\star \\
NAxiome & ::= & [A-Z][A-Za-z0-9]^\star \\
NTerminal & ::= & [a-z][A-Za-z0-9]^\star \\
V & ::= & Var \mid Val \\
Terminal & ::= & NTerminal\ (\ ExPar^\star\ ) \\
OpCond & ::= & = \mid > \\
Cond & ::= & V \quad OpCond \quad V \mid \textsf{true} \\
OpPar & ::= & + \mid - \mid x \\
ExPar & ::= & (\ ExPar \quad OpPar \quad ExPar\ ) \\
 & \mid & V \\
Axiome & ::= & NAxiome\ (\ ExPar^\star\ ) \\
Terme & ::= & Axiome \quad Terme \\
 & \mid & Terminal \quad Terme \mid \epsilon \\
RWrule & ::= & Axiome : Cond \rightarrow Terme
\end{array}
$$

FIG. 4.23 – Proposition de grammaire pour les règles de réécriture. $A^\star$ désigne une liste d'éléments de type A.

4.4.3 La sémantique utilisée

Pour pouvoir utiliser un cache pour les *FL-systems*, nous avons besoin d'une réécriture déterministe qui peut être obtenue grâce à une sélection des règles de réécriture déterministe. Nous proposons ici une formalisation de la sémantique utilisée pour la réécriture des *FL-systems*. La sémantique décrit la façon dont sont traitées les règles lors de la réécriture. Tout d'abord, nous définissons $\mathcal{R}$ comme la liste (ordonnée) des règles de réécriture d'un *FL-system* donné. Soit $\mathbf{A}(v^*)$ un axiome tel que $v^* \in \mathbb{R}^*$. Nous avons alors la sémantique donnée en figure 4.24.

$$
\frac{x \in \text{Terminaux}^\star \qquad \mathbf{A}(p^*) : c_* \wedge c \rightarrow \boldsymbol{T} \ = \ \texttt{first}(\mathcal{R}, \mathbf{A}(v^*))}{x\ \mathbf{A}(v^*)\ y \quad \rightarrow_{\mathcal{R}} \quad x\ \texttt{eval}(\boldsymbol{T}[v^*/p^*])\ y}
$$

FIG. 4.24 – Sémantique des règles de réécriture

Nous définissons $\texttt{first}(\mathcal{R}, \mathbf{A}(v^*))$ la fonction qui retourne la règle de réécriture "$\mathbf{A}(p^*) : c_* \wedge c \rightarrow \boldsymbol{T}$" où :

- "$\mathbf{A}(p^*) : c \rightarrow \boldsymbol{T}$" = r_i est la première règle de $\mathcal{R}$ s'appliquant (*i.e.* la condition "c" est vraie) à $\mathbf{A}(v^*)$

– c_* est la conjonction des négations des condition des règles précédant r_i dans $\mathcal{R}$ et s'appliquant à un axiome de nom **A**

$X[a/b]$ retourne le résultat de la substitution dans X de toutes les occurrences de b par a. Ainsi, $(\mathbf{A}(x+3,y))[(1,0)/(x,y)]$ rend $\mathbf{A}(1+3,0)$. La fonction `eval`($\boldsymbol{T}$) parcourt le terme $\boldsymbol{T}$ et évalue les expressions y apparaissant pour obtenir les paramètres numériques de l'étape de réécriture suivante. Par exemple, `eval`$(\mathbf{A}(1+3,0))$ retourne $\mathbf{A}(4,0)$.

Exemple 4.4.1: Un exemple de *FL-system*

$$\mathbf{A}(m,n) : m = 0 \rightarrow \epsilon$$
$$\mathbf{A}(m,n) : n > 0 \rightarrow \mathbf{A}(m,n-1)$$
$$\mathbf{A}(m,n) : n = 0 \rightarrow \epsilon$$

Étant donné le *FL-system* de l'exemple 4.4.1, l'axiome $\mathbf{A}(1,2)$ se réécrit en $\mathbf{A}(1,1)$ avec `first`$(\mathcal{R}, \mathbf{A}(1,2))$ égal :

$$\mathbf{A}(m,n) : \neg(m=0) \wedge (n>0) \rightarrow \mathbf{A}(m,n-1)$$

Dans cet exemple, la condition $m \neq 0 \wedge n > 0$ donne la dépendance vis-à-vis des variables m et n de la règle appliquée. Si la négation de $m = 0$ n'était pas présente, l'importance de la valeur de m ne serait pas explicite. Toujours dans ce même exemple, la fonction `eval` remplace le paramètre m par sa valeur (1), et évalue l'expression "$n-1$" à 1.

Le paragraphe précédent aborde la dépendance qui existe entre la valeur des paramètres d'un axiome et le résultat final de sa réécriture. La section suivante propose d'aller plus loin dans l'étude de cette dépendance.

4.4.4 Calcul dynamique des dépendances

Le système de cache présenté dans cette section met à profit la propriété de *joignabilité* de certains termes dans un système de réécriture [DJ90]. Si deux termes se réécrivent en un même terme sous forme normale (*i.e.* composé uniquement de terminaux) alors ces deux termes sont dit *joignables*. Dans cette section, nous exposons la méthode employée afin de calculer dynamiquement un ensemble, aussi grand que possible, de termes joignables avec le terme en cours de réécriture. La méthode employée s'inspire de travaux sur la confidentialité de programmes séquentiels [LGJ05]. Elle se base sur la modification de la sémantique de réécriture afin de manipuler des *valeurs étiquetées*. La réécriture d'un axiome, paramétré par des *valeurs étiquetées*, retourne alors un terme sous forme normale, lui-même marqué par une étiquette. Cette dernière découle des étiquettes des paramètres de l'axiome qui ont influencé cette réécriture. Les travaux de cette section sont principalement le fruit des travaux de Gurvan Le Guernic. Néanmoins, il est important de les présenter brièvement ici pour comprendre le fonctionnement du système de cache proposé.

4.4.4.1 Adaptation de la grammaire

La simple modification, donnée en figure 4.25, de notre grammaire initiale (figure 4.23) permet d'intégrer les valeurs étiquetées. $\mathcal{P}(\mathbb{L}abel)$ est l'ensemble des en-

sembles de $\mathbb{L}abel$. Les valeurs (Val) sont maintenant des paires notées "$e : n$". Le deuxième élément de cette paire (n), aussi appelé valeur numérique, appartient à $\mathbb{R}^+$. Le premier élément (e) est un ensemble de *labels* formant une *étiquette*. Chaque étiquette de l'axiome initial[3] est composée d'un unique label. Chacun de ces labels n'est utilisé qu'une seule fois (*cf.* exemple 4.4.2).

Exemple 4.4.2: Exemple d'étiquetage initial
$\mathbf{A}(\{l1\} : 7, \{l2\} : 22.5, \{l3\} : 10, \{l4\} : 1, \{l5\} : \textit{green})$

$$\mathbb{E}tiquette = \mathcal{P}(\mathbb{L}abel)$$
$$Val = (\mathbb{E}tiquette \times \mathbb{R}^+)$$

FIG. 4.25 – Grammaire des valeurs étiquetées

4.4.4.2 Une nouvelle sémantique

La figure 4.26 présente la nouvelle sémantique utilisée pour la réécriture de termes dont les valeurs sont étiquetées. Cette sémantique présente de grandes similitudes avec celle présentée dans la section 4.4.3. Il est facile de se convaincre que, mis à part les étiquettes, ces deux sémantiques réécrivent un axiome de la même façon.

$$\frac{\begin{array}{c} x \in \text{Terminaux}^\star \\ \mathbf{A}(p^*) : c_* \wedge c \rightarrow \boldsymbol{T} \;=\; \texttt{first}(\mathcal{R}, \mathbf{A}(v^*)) \\ e_c = (\texttt{var}(c_* \wedge c))[\texttt{lab}(v)^*/p^*] \\ e_{op} = (\texttt{var}(\texttt{op}(\boldsymbol{T})))[\texttt{lab}(v)^*/p^*] \\ e' = e \cup e_c \cup e_{op} \end{array}}{e :\ x\ \mathbf{A}(v^*)\ y \quad \rightarrow_{\mathcal{R}} \quad e' :\ x\ \texttt{eval}(\boldsymbol{T}[v^*/p^*])\ y}$$

FIG. 4.26 – Nouvelle sémantique des règles de réécriture

Dans cette nouvelle sémantique, chaque terme est également étiqueté. Cette étiquette identifie l'ensemble des paramètres qui ont influencé la réécriture. Afin de calculer la nouvelle étiquette d'un terme lors de sa réécriture, nous introduisons trois nouvelles fonctions :

`op` retourne la liste des opérations contenues dans son argument. Par exemple, "$\texttt{op}(\mathbf{A}(x, x+y, x+1))$" rend la liste $[x+y, x+1]$.

`var` retourne l'ensemble des variables de son argument. Ainsi "$\texttt{var}(x > y)$" rend en résultat l'ensemble $\{x, y\}$.

`lab` extrait l'étiquette d'une valeur. Par exemple, "$\texttt{lab}(e : 3)$" retourne l'étiquette e.

La fonction `eval` est également modifiée afin de calculer avec des valeurs étiquetées. Lors de l'évaluation d'un paramètre sous forme d'opération, `eval` fait abstraction des étiquettes et effectue l'opération à partir des valeurs numériques. Une nouvelle étiquette vide est attribuée au résultat. Par exemple, $\texttt{eval}(\mathbf{A}((e:1) + 3))$ retourne $\mathbf{A}(\emptyset : 4)$.

Lors de la réécriture d'un terme t dans un terme t' ($e : t \rightarrow_{\mathcal{R}} e' : t'$), la nouvelle étiquette (e') est obtenue en faisant l'union des labels de l'ancienne étiquette avec les labels des étiquettes e_c et e_{op}, où :

[3] On appelle axiome initial, l'axiome servant de point de départ à la réécriture.

e_c contient les labels des paramètres influençant la sélection de la règle de réécriture pour cette étape de l'évaluation.

e_{op} contient les labels des paramètres qui sont intervenus dans le calcul d'une opération à cette étape.

4.4.4.3 Interprétation des étiquettes résultants de la réécriture d'un axiome

L'exemple 4.4.3 présente l'ensemble $\mathcal{R}$ des règles de réécriture qui sont utilisées dans cette partie.

Exemple 4.4.3: Ensemble des règles des axiomes **A** et **B**

$\mathbf{A}(c, d, x) : c = 0 \rightarrow \mathbf{B}(d, x)$
$\mathbf{A}(c, d, x) : c > 0 \rightarrow \mathsf{v}(d + c)\ \mathbf{A}(c - 1, d, x)$
$\mathbf{B}(d, x) : d = 0 \rightarrow \mathsf{t}()$
$\mathbf{B}(d, x) : d > 0 \rightarrow \mathsf{u}(x)\ \mathbf{B}((d - 1), x)$

Soit e l'étiquette associée au terme sous forme normal résultant de la réécriture de l'axiome **C** ($\emptyset$:$\mathbf{C}(v^*) \rightarrow^{\dagger}_{\mathcal{R}} e$:***NT***). e contient les labels de tous les paramètres qui ont influencés la sélection des règles lors de la réécriture de **C** ; et donc la structure générale[4] du terme sous forme normale obtenu. Par exemple, l'évaluation de l'axiome $\mathbf{B}(e_1{:}0, e_2{:}3)$ extériorise[5] uniquement l'étiquette de son premier paramètre et rend le terme sous forme normale $\mathsf{t}()$. Celui-ci n'ayant pas de terminaux avec paramètre, le corollaire de reconstructibilité que nous avons proposé (voir [LGP05]) implique que tout axiome de la forme $\mathbf{B}(\_{:}0, \_)$ se réécrit sous forme normale en $\mathsf{t}()$. La réécriture de l'axiome $\mathbf{A}(e_1{:}0, e_2{:}0, e_3{:}3)$ fonctionne de façon similaire. Le résultat de sa réécriture est donné dans l'exemple 4.4.4. L'interprétation du résultat de l'évaluation de l'axiome $\mathbf{A}(e_1{:}1, e_2{:}1, e_3{:}1)$ est plus complexe. Le résultat, donné dans l'exemple 4.4.4, montre l'extériorisation des étiquettes des deux premiers paramètres uniquement. Il est donc possible d'en déduire que la réécriture sous forme normale de tout axiome de la forme $\mathbf{A}(\_{:}1, \_{:}1, \_)$ rend un terme ayant pour structure générale $\mathsf{v}(\_)\ \mathsf{u}(\_)\ \mathsf{t}()$. Plus précisément, ce même corollaire implique que, le paramètre du terminal v ayant une étiquette vide, le résultat a pour forme $\mathsf{v}(\_{:}2)\ \mathsf{u}(\_)\ \mathsf{t}()$. De plus, étant donné que le paramètre de u a pour étiquette celle du troisième paramètre de l'axiome initial, ce même corollaire implique que la réécriture sous forme normale de tout axiome de la forme $\mathbf{A}(\_{:}1, \_{:}1, \_{:}X)$ rend un terme ayant pour forme $\mathsf{v}(\_{:}2)\ \mathsf{u}(\_{:}X)\ \mathsf{t}()$.

---

[4] *i.e.* abstraction faite des paramètres des terminaux.

[5] On dit qu'une réécriture extériorise une étiquette si cette étiquette se retrouve dans l'étiquette du terme sous forme normale obtenu.

Exemple 4.4.4: Résultats d'évaluations
Le tableau suivant présente des exemples d'évaluation des axiomes **A** et **B** de l'exemple 4.4.3.

Axiome	Résultat
$\mathbf{B}(e_1 : 0, e_2 : 3)$	$e_1 : [\mathsf{t}()]$
$\mathbf{A}(e_1 : 0, e_2 : 0, e_3 : 3)$	$\{l_1, l_2\} : [\mathsf{t}()]$
$\mathbf{A}(e_1 : 1, e_2 : 1, e_3 : 1)$	$\{l_1, l_2\} : [\mathsf{v}(\emptyset : 2); \mathsf{u}(e_3 : 1); \mathsf{t}()]$

La sémantique proposée permet ainsi, en apposant une étiquette aux règles et à leurs paramètres, de déterminer quels paramètres sont intervenus dans le résultat d'une réécriture.

4.4.5 Application à un système de cache

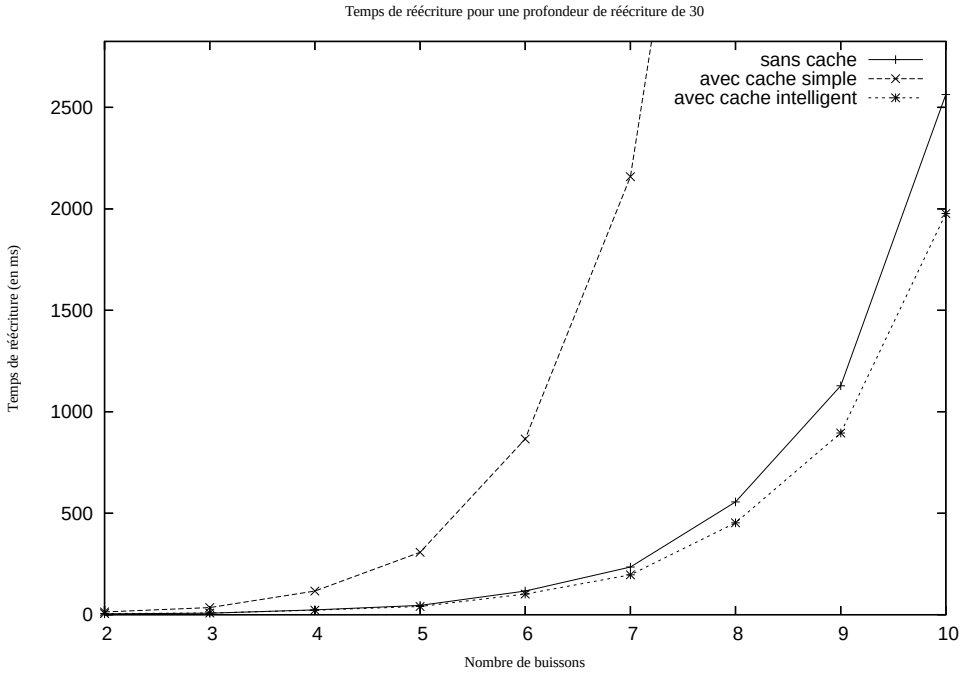
Le *cache* (ou mémoire cache) est une mémoire intermédiaire entre le disque (les données) et le processeur. En stockant les informations les plus susceptibles d'être demandées par le processeur, il permet d'accélérer les communications entre ce dernier et le disque, et par conséquent les performances. Dans notre contexte, le cache contient les résultats de processus de réécriture : les valeurs. La clef à laquelle est associée chaque valeur est l'axiome qui l'a produit.

Dans un cache standard, la paire (clef,valeur) générée par une réécriture est très simple. La clef correspond à l'axiome initial. Elle inclut le nom de l'axiome et la valeur effective de ses paramètres. La valeur associée à cette clef est le terme normal obtenu lors de la réécriture. Cependant, il arrive qu'un paramètre n'intervienne pas dans la réécriture de l'axiome, soit pour une raison de niveau de détail pris en compte, soit parce que le paramètre intervient uniquement au moment du rendu de l'environnement virtuel. Ainsi, avec les règles proposées dans l'exemple 4.4.3 et comme indiqué dans la section 4.4.4.3, les axiomes $\mathbf{B}(0, 3)$ et $\mathbf{B}(0, 0)$ se réécrivent de la même façon. En utilisant un calcul de dépendance comme celui qui est présenté en section 4.4.4, il est possible d'améliorer la couverture de la clef de façon à ce qu'une concordance soit déclenchée dans le cache pour deux axiomes qui ne diffèrent que par la valeur de paramètres n'intervenant pas dans la réécriture de ces axiomes. Pour les détails de la gestion des paires (clefs,valeur) du système de cache, on se référera à [LGP05].

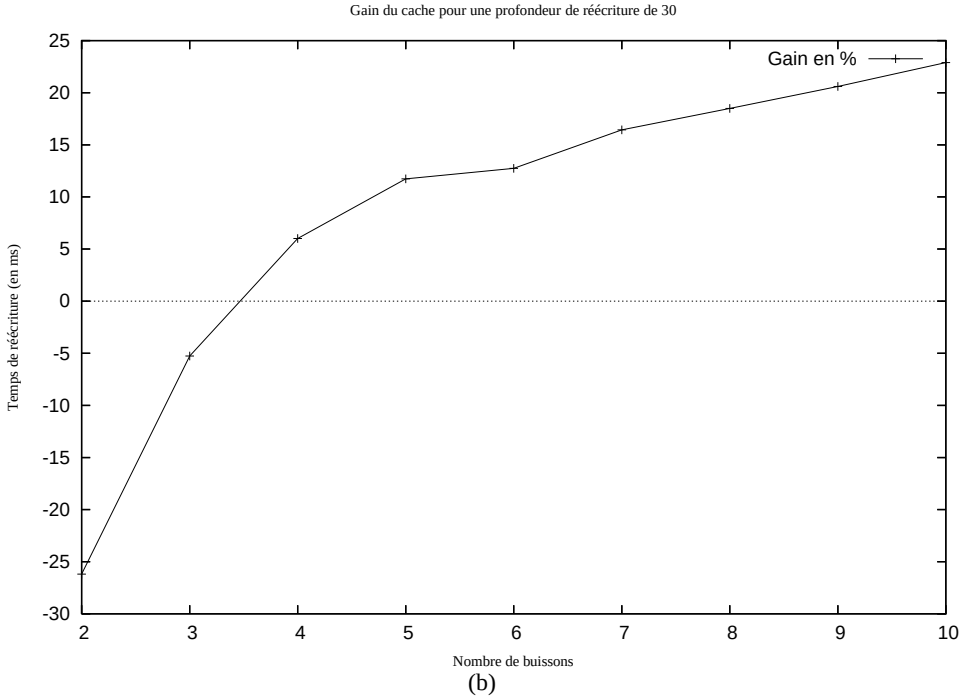
4.4.6 Résultats expérimentaux

Un prototype *FL-system* a été implémenté en *Prolog*. Il simule la réécriture d'un terme sans cache, avec un cache simple, ou avec le système de cache proposé dans cette section. Chaque clef du cache simple se compose du nom de l'axiome et des valeurs effectives des paramètres de l'axiome ayant servi à la génération de cette clef. Nos résultats sont illustrés par la figure 4.27. Nous avons constaté que, généralement, plus la profondeur augmente, plus notre système de cache s'avère efficace (pour converger aux alentour de 25% par rapport à la réécriture sans cache dans le cadre de l'étude). En effet, par rapport à la réécriture sans cache, la méthode proposée ici profite alors de l'augmentation du nombre de termes *joignables*[6] rencontrés lors

[6]Deux termes sont dits *joignables* si leur réécriture aboutit au même terme sous forme normale



(a)



(b)

FIG. 4.27 – Résultats expérimentaux du cache proposé. (a) : temps de réécriture obtenu en fixant la profondeur de la réécriture à 30 et en faisant varier le nombre de buissons. On remarque qu'un cache simple est moins efficace que la réécriture sans cache. (b) : Gain obtenu par notre cache.

de la réécriture. Pour le détail des tests effectués et l'étude des temps de réécriture en fonction de la complexité du *FL-system* se référer à [LGP05].

4.4.7 Conclusion

Dans cette section, nous avons présenté un système de cache *intelligent* basé sur un calcul dynamique des dépendances. Ce nouveau système de cache pour les *FL-systems* permet une abstraction dans le processus de modélisation géométrique : la réutilisation des objets géométriques, jusqu'ici explicite, devient implicite. Cette technique est plus efficace qu'un cache simple grâce à la plus grande généricité des paires (clef,valeur) produites. Il est ainsi possible de générer la forme normale d'un axiome qui n'a pas encore été rencontré par instanciation de la valeur associée à une clef *proche* de l'axiome à évaluer. Néanmoins, il est important de ne pas calculer des paires (clef,valeur) pour chaque règle de réécriture, sous peine de faire déborder le cache. Une solution à ce problème consiste à identifier, lors de l'écriture d'un *FL-system* ou automatiquement par analyse, les règles les plus utilisées.

Notre système de cache permet une plus grande souplesse dans la modélisation d'environnements très complexes. En effet, considérons un *FL-system* représentant un arbre et possédant un paramètre $color$ stipulant la couleur des feuilles de celui-ci. Une fois cet arbre utilisé pour modéliser une forêt, la modification de la valeur du paramètre $color$ pourra être prise en compte de façon automatique par le cache. L'objet représentant une feuille sera alors modifié pour prendre en compte le changement de couleur, et le reste de la forêt changera de couleur par la même occasion, sans modifier le cache et sans nécessiter la moindre réécriture.

Enfin, le *FL-system Intelligent Cache* constitue une optimisation complémentaire à l'instanciation approximative [DHL+98]. En effet, notre technique permet de déterminer quels sont les termes dont les réécritures sont identiques, tandis que l'instanciation approximative permet de regrouper des termes similaires grâce à leur proximité dans l'espace des valeurs des paramètres. Notre technique permettrait donc d'améliorer le *clustering* en identifiant les paramètres qui n'influencent pas la réécriture. En effet, ceci revient à réduire la dimension de l'espace des paramètres et, de ce fait, à faciliter le *clustering*. Le couplage de ces deux techniques constitue donc une perspective intéressante pour nos prochains travaux.

4.5 Discussion

Dans ce chapitre, nous avons présenté les *FL-systems* ainsi que des applications de ces systèmes à la modélisation d'environnements urbains. Les *FL-systems* étendent les *L-systems* par l'utilisation de fonctions comme terminaux. Nous avons par ailleurs proposé deux stratégies de réécriture pour ces systèmes, la possibilité de générer un modèle géométrique au fil de la réécriture, ainsi qu'un système de cache visant à améliorer les performances de la réécriture. Ainsi, nous pensons avoir montré

la capacité de ces systèmes à représenter efficacement des objets géométriques complexes présentant une forte auto-similarité. Cette efficacité concerne notamment leurs propriétés de compression de la représentation géométrique, leur utilisation pour la transmission de modèles complexes à travers un réseau et la navigation en temps réel.

Par ailleurs, ces systèmes permettent l'incrustation d'informations à l'intérieur des modèles générés. En effet, les modèles géométriques ne représentent qu'une partie des structures pouvant être engendrées. On pourrait, par exemple, attacher des informations sémantiques aux objets et structures générées pour leur utilisation dans divers contextes comme l'animation comportementale. De même, en effectuant une réécriture continue, de tels modèles peuvent être utilisés pour l'animation procédurale des objets manipulés. La plus grande qualité des *FL-systems* est finalement leur adaptabilité. De nombreuses perspectives restent donc à explorer.

Néanmoins, les *FL-systèmes* restent difficiles à manipuler. D'une part le formalisme est complexe et nécessite un utilisateur expérimenté. D'autre part, leur intégration à une plate-forme de navigation temps-réel devient complexe dès lors que la réécriture elle-même doit être contrôlée. En effet, une réécriture paresseuse des *FL-systems* présenterait de nombreux avantages : nous pouvons par exemple imaginer le déclenchement de la réécriture sur des critères de perception visuelle. Nous n'avons malheureusement pas approfondi la question.

Pour finir, nous estimons que, si l'utilisation de *L-systems* ou de *FL-systems* est intéressante pour la modélisation de plantes et de bâtiments, leur utilisation pour modéliser le réseau urbain reste inappropriée. En effet, il est difficile de contrôler le modèle engendré : la propriété d'amplification des données, qui donne leur expressivité à ces systèmes, limite leur utilité dans la modélisation interactive d'environnements aussi complexes. En effet, la modification minime de paramètres initiaux a des conséquences sur l'ensemble du modèle engendré. Dans le cadre de la modélisation interactive d'environnements urbains, il faudra développer des méthodes de modification locale du réseau urbain. Par ailleurs, la modélisation de nombreux éléments urbains est difficile à mettre en place en utilisant un tel formalisme. On pourra penser à la décomposition des îlots en parcelles par exemple. C'est pour répondre à ces difficultés que nous présentons, dans le chapitre suivant, une nouvelle méthode de modélisation urbaine offrant une vision plus systémique du problème.

Chapitre 5

Vers une modélisation interactive de la ville

L'histoire nous montre la tendance de l'esprit vers le simple. Le simple est l'effet du jugement, du choix ; il est le signe de la maîtrise. S'arrachant aux complexités, on inventera les moyens qui manifesteront l'état de conscience.

Le Corbusier, [Cor30]

Il existe, à ce jour, peu de méthodes appropriées à la modélisation de la ville. En effet, les méthodes traditionnelles de modélisation géométrique et les logiciels les implémentant n'ont pas été conçus en prenant en compte les contraintes propres aux modèles tels que les villes. En particulier, la taille des modèles impliqués, en terme de complexité, rend difficile voire impossible leur visualisation ; d'une part pour des raisons de performances des techniques de visualisation, d'autre part pour des problèmes de lisibilité. Ainsi, même lorsque l'on peut observer un modèle de ville, il contient souvent trop de données pour que nous puissions les interpréter et manipuler efficacement le modèle dans son ensemble. Trois problèmes complémentaires sont ainsi impliqués : la modélisation en tant que processus, la représentation du modèle et sa visualisation. Ces trois problèmes sont bien enchevêtrés puisque la modélisation suggère l'existence d'une représentation préalable du modèle (cf. figure 5.1). Il en est de même pour la visualisation. Le problème de la représentation du modèle semble ainsi central et nous le traitons dans ce chapitre de façon prioritaire.

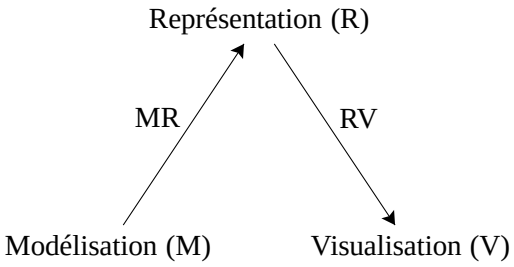


FIG. 5.1 – Le triangle Modélisation / Représentation / Visualisation.

La représentation hiérarchique utilisée pour modéliser la ville est présentée en section 5.1. L'architecture d'une méthode de modélisation s'appuyant sur cette représentation est ensuite décrite en section 5.2. Le processus de modélisation ainsi que les algorithmes sur lesquels il s'appuie sont proposés en section 5.3, suivis de quelques résultats 5.4 et finalement d'une conclusion et d'une discussion en section 5.5.

5.1 Représentation hiérarchique

L'architecture c'est "mettre en ordre". Mettre en œuvre quoi ? Des fonctions et des objets. Occuper l'espace avec des édifices et des routes. Créer des vases pour abriter les hommes et créer des communication utiles pour s'y rendre.

Le Corbusier [Cor30]

La ville est un objet complexe qu'il est difficile d'appréhender dans sa globalité. Certes, le problème du recueil des données, abordé en section 2.1, est un des

facteur de cette difficulté. Néanmoins, un facteur plus décisif encore réside dans la quantité et la densité d'informations et d'objets dont une ville est constituée. Cette complexité incite à utiliser différents niveaux de détails pour étudier la ville. De fait, la nature de l'acteur effectuant l'étude ou la modélisation lui impose un niveau de détail, une échelle propre à son champ d'étude : les géographes en étudient la démographie et la morphologie, les urbanistes l'aménagement, les architectes le peuplement, ...C'est en effet une nécessité, pour que l'esprit puisse analyser un objet d'une telle complexité, de le simplifier, de le découper, d'en analyser chaque morceau pour les recoller ensuite. C'est pourquoi nous définissons une hiérarchie d'abstraction pour représenter la ville.

5.1.1 abstraction

Nous définissons ici une hiérarchie d'abstraction pour la ville inspirée des études sur la morphologie urbaine d'Allain [All04] et de Lynch sur l'image mentale de la cité [Lyn60]. En effet, d'Allain, nous allons utiliser les niveaux de la hiérarchie : la macroforme, l'îlot, la parcelle, le bâtiment, l'espace public. De Lynch, nous utilisons les éléments de la cité et leurs influences respectives : le chemin, les limites, les nœuds (en laissant de côté les points de repère et les quartiers pour l'instant).

Habraken [Hab00] propose quant à lui plusieurs notions intéressantes concernant les interactions des acteurs intervenant dans la construction et l'évolution des environnements urbains construits. En effet, il étudie notamment les notions de contrôle et de dominance entre les différents niveaux de la hiérarchie que constitue la ville. Il définit ainsi les éléments et combinaisons d'éléments dont est constitué l'environnement construit comme des *configurations* sur lesquelles des *agents* exercent un *contrôle*. Les interactions des agents et des configurations créent ainsi des situations où certaines configurations en dominent d'autres, engendrant ainsi des *hiérarchies de dominance*. En effet, de cette façon, le réseau urbain domine les bâtiments puisque la modification du premier peut engendrer la modification du second (dans le cas d'élargissement de voie par exemple), l'inverse étant beaucoup plus rare. Ce type de dominance est appelé *encadrement*[1]. Les *hiérarchies d'assemblage* représentent un autre type important de hiérarchies de dominance reflétant les dépendances résultant d'un processus de construction. De cette façon, du fait de la gravité, les murs dépendent des fondations sur lesquelles ils sont construits. Pour finir, les *hiérarchies de territoire* se basent sur le principe d'inclusion : dans de telles hiérarchies, "la dominance est exprimée en refusant l'accès au territoire inclus." Ainsi, il vous est possible d'interdire l'accès à votre logis à quiconque (ou presque). La figure 5.2 illustre une hiérarchie de territoire, trois niveaux d'encadrement et un diagramme d'assemblage.

5.1.2 Réseau urbain

La ville est tout d'abord un réseau, un réseau urbain fait d'autoroutes, de routes, d'avenues, de boulevards, de ruelles, d'impasses, de carrefours, de places. Ce réseau est constitué d'espace navigables, de *voies*. Ces voies sont connectées par leurs *intersections*. Les régions closes entourées par des voies ou des limites sont appelées *îlots* (voir figure 5.3(a)).

Ce niveau d'abstraction correspond au niveau du plan, de la trame. Le graphe ainsi définit est planaire, néanmoins, il repose sur un *terrain* qui lui, n'est pas un

[1] encadrement est une traduction personnelle du mot *enclosure*

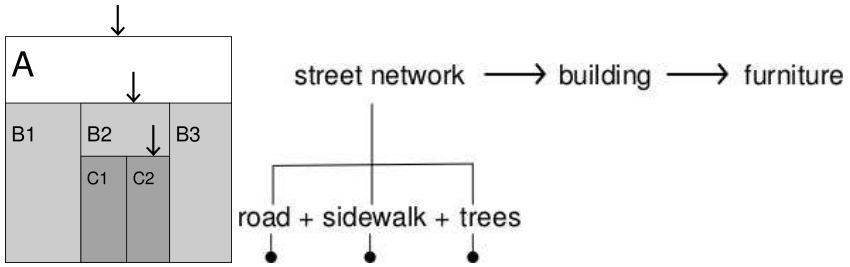


FIG. 5.2 – A gauche : une hiérarchie de territoire. C2 est inclus dans B2, lui-même inclus dans A. Les flèches illustrent les frontières à franchir pour passer d'un niveau de la hiérarchie à l'autre, i.e. la profondeur de territoire. A droite : trois niveaux d'encadrement et un diagramme d'assemblage [Hab00].

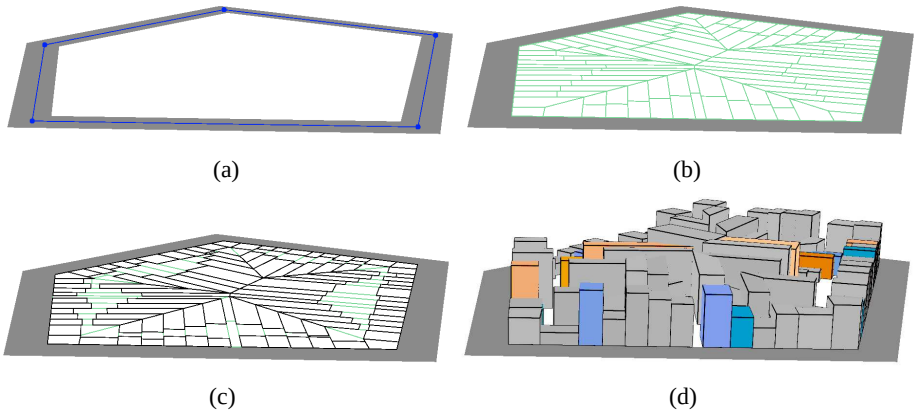


FIG. 5.3 – Les différents niveaux de la hiérarchie du réseau urbain. (a) : L'îlot. (b) : La parcelle. (c) : Le cadastre. (d) : Le bâtiment.

plan. Le terrain contient lui aussi un graphe représentant les limites naturelles au développement de la ville. Ce graphe est fait d'intersections et de *limites*.

Du point de vue de la théorie des graphes, ce graphe constitue ainsi un graphe planaire acyclique non-orienté. Dans ce graphe, les faces sont les régions entourées d'arêtes et à l'intérieur desquelles ne réside aucun sommet.

Les voies, intersections et îlots possèdent deux niveaux d'abstraction. Au plus haut niveau, topologique, une intersection est un point, une voie un segment, un îlot une face. Au niveau le plus bas, géométrique, voies, intersections et îlots sont des faces qui partagent leurs contours. Ces trois types d'objets recouvrent la totalité de la surface construite de la ville. Leur agrégation forme la *macroforme*.

Cette abstraction définit une généralisation.

5.1.3 îlot et parcelle

Les éléments du réseau urbain sont décomposés, au niveau d'abstraction inférieur, en sous-éléments :

- une voie est décomposée en *trottoir*, *pavé* et éventuellement *terre-plein*.
- une intersection est décomposée en *trottoir*, *pavé* et éventuellement *rond-point*.
- un îlot est décomposé en *parcelles*.

Dans ce chapitre, nous ne traitons que la décomposition des îlots en parcelles. Cette abstraction défini une agrégation.

5.1.4 cadastre

Le niveau d'abstraction suivant ne concerne que les parcelles : en effet, les parcelles sont décomposées en *empreintes de bâtiments*, et *jardins*. Il s'agit encore ici d'agrégation.

5.2 Architecture de la méthode de modélisation

L'architecture de la méthode de modélisation proposée peut être synthétisée par la figure 5.4. Le processus de modélisation peut être découpé en différentes étapes correspondant aux différents niveaux d'abstraction de la ville. En entrée de la méthode, nous avons des images, permettant la création de la carte de terrain ainsi que la détermination des limites naturelles au développement urbain (cours d'eau par exemple). D'autres images pourront aussi servir à déterminer la hauteur des bâtiments par exemple.

Dans la figure 5.4, nous pouvons distinguer à gauche le processus de modélisation sur lequel l'utilisateur peut intervenir à tout moment, et, à droite, la structure de données utilisée pour représenter la ville. Il existe une forte correspondance entre les niveaux respectifs du processus et du modèle. En effet, nous pensons que c'est en s'appuyant sur nos connaissances de l'objet modélisé, ici la ville, que nous pouvons concevoir les méthodes de modélisation les plus efficaces.

Pour finir, cette figure contient l'utilisation de géométrie procédurale à partir des objets produits par la méthode. Néanmoins, cette utilisation n'est pas traitée dans ce chapitre et illustre seulement notre volonté de fusionner ces travaux et ceux présentés dans le chapitre précédent.

En ce qui concerne l'implémentation de nos algorithmes, décrits en section 5.3, nous utilisons la bibliothèque *CGAL*[2] qui nous permet de définir des algorithmes génériques en utilisant des noyaux de calcul filtrés. Ces noyaux permettent notamment d'effectuer les calculs les plus sensibles (comme les intersection par exemple) de façon exacte en utilisant des nombre de précision quelconque, et d'accélérer les calculs où la précision est peu importante en utilisant des nombres à virgule flottante (*float*) par exemple.

5.3 Modélisation

À partir de la représentation hiérarchique proposée, nous présentons ici notre méthode de modélisation hiérarchique de la ville. Cette méthode nous permet de tirer parti de la représentation présentée en travaillant sur chaque niveau d'abstraction indépendamment ou presque.

La méthode proposée est hybride. En effet, il est possible de modéliser une ville de façon purement procédurale. Néanmoins, nous avons introduit la possibilité d'intervenir de façon interactive dans le processus de modélisation. C'est là, justement,

[2]pour Computational Geometry Algorithms Library ou bibliothèque d'algorithmes de géométrie algébrique

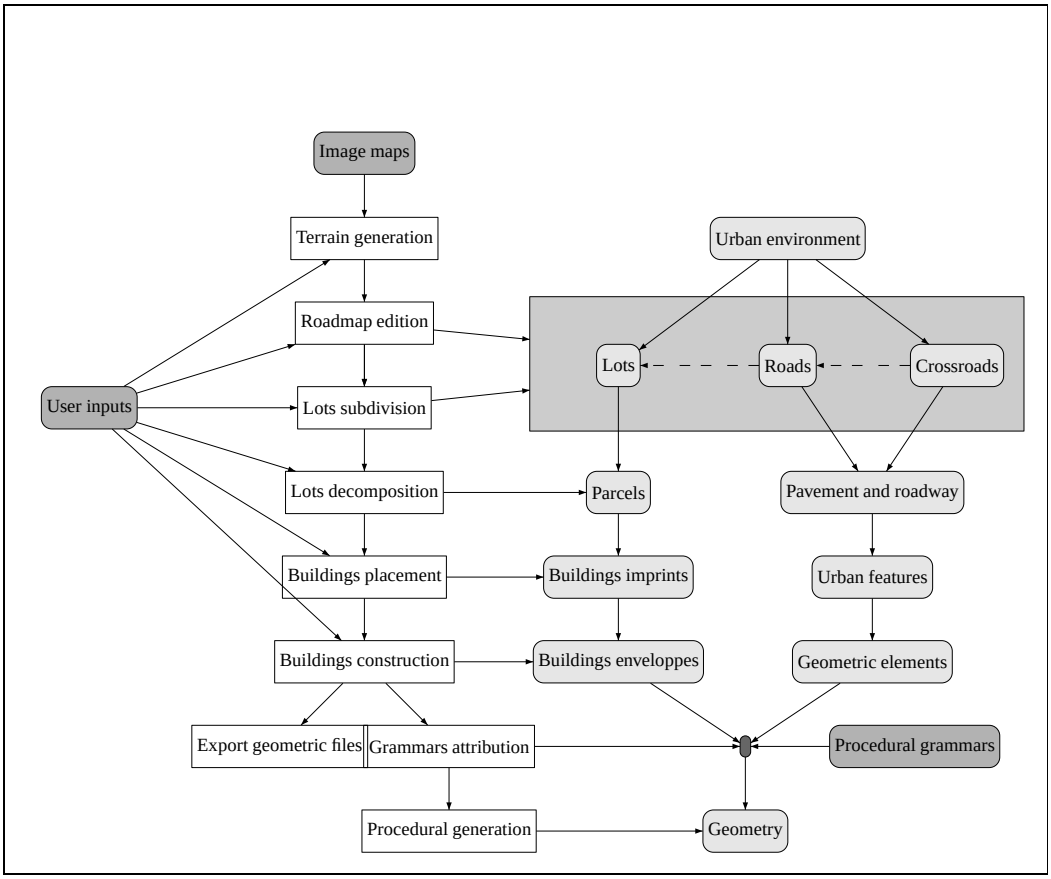


FIG. 5.4 – Architecture du processus de modélisation d'environnements urbains proposé. En gris foncé : les données et entrées fournies par l'utilisateur. En gris clair : la structure de données hiérarchique construite par le logiciel. À l'intérieur de cette structure de données, les îlots, voies et nœuds ont été regroupés pour illustrer l'influence des étapes d'édition des voies et de subdivision des îlots sur l'intégralité de ce niveau d'abstraction. Pour finir, les flèches en pointillés indiquent des relations de domination entre les éléments d'un même niveau d'abstraction.

que les méthodes procédurales prennent, à notre avis, tout leur sens : en tant qu'outils dont on maîtrise la manipulation, en tant que technique complémentaire aux techniques traditionnelles.

Pour présenter notre méthode, nous allons suivre les différentes étapes du processus de modélisation que nous venons de présenter, en commençant par la création du terrain (section 5.3.1), du réseau urbain (section 5.3.2), de la décomposition des îlots (section 5.3.3), et en finissant par la création du cadastre (section 5.3.4).

5.3.1 Terrain

La première étape de la modélisation d'une ville consiste à choisir son emplacement : le terrain. La topographie du terrain est assimilée ici à une carte du relief sous la forme d'une image. Dans cette image, nous utilisons chaque composante de couleur pour une composante du terrain :

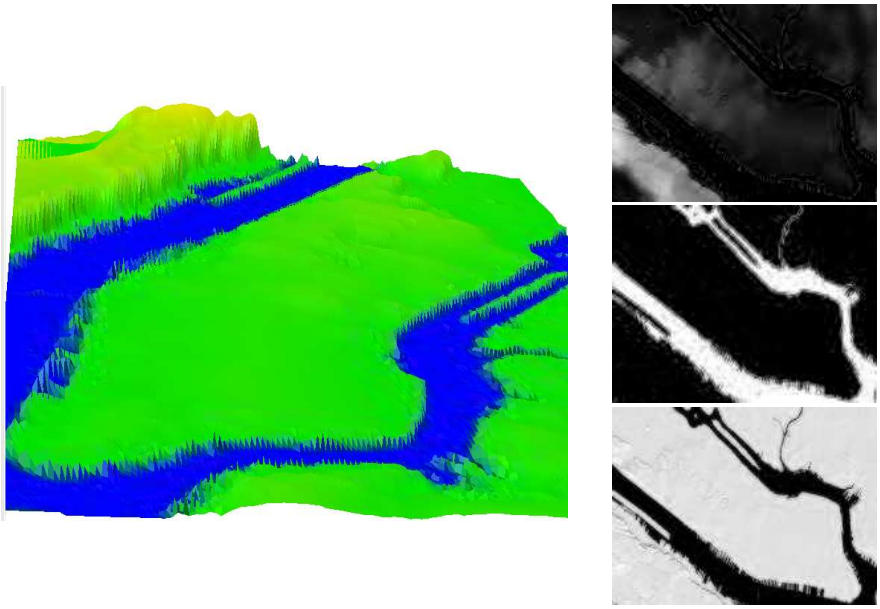


FIG. 5.5 – À gauche : Terrain construit à partir d'une image satellite de Manhattan. Ici, le facteur d'échelle utilisé pour le relief a été délibérément exagéré. À droite : les trois composantes de la carte, respectivement rouge, bleue et verte.

- La composante rouge détermine la hauteur du terrain au pixel donné.
- La composante verte détermine la densité de végétation.
- La composante bleue marque les zones aquatiques (océan, mer, fleuve, rivière, lac...).

Lors du chargement de la carte de terrain, il faut par ailleurs spécifier :

- L'échelle de la carte sur les deux axes, c'est à dire ici la taille en mètres à laquelle correspond chaque pixel de l'image.
- Le facteur d'échelle pour le relief, c'est à dire les hauteurs minimales et maximales utilisées dans la carte, de façon similaire aux cartes *USGS*[3]. Ces deux hauteurs h_{min} et h_{max} correspondent respectivement aux valeurs 0 et 255 de la composante rouge de la carte. Ceci nous permet ensuite d'interpoler toutes les autres valeurs pour une composante de valeur c, on a alors $h = h_{min} + c \times (h_{max} - h_{min})$.

À partir de la carte et des paramètres associés, nous construisons une grille, ainsi que le réseau correspondant aux limites naturelles que le terrain offre. Le résultat de la modélisation du terrain est illustré par la figure 5.5.

5.3.2 Réseau urbain

Topologie

Nous avons mis en place une interface simple qui permet de dessiner un réseau urbain. Cette interface est similaire à une interface de dessin vectoriel : il s'agit ainsi de placer les segments de voirie. Des outils simples de suppression et de découpage

[3]U.S. Geological Survey, organisme en charge de l'étude du paysage, des ressources naturelles, etc.

des segments ajoutés permettent une plus grande précision. Par ailleurs, les intersections entre les segments insérés sont calculées automatiquement et l'utilisateur peut sélectionner les intersections et extrémités des segments existants et s'y raccrocher pour ajouter de nouveaux segments. De plus, en sélectionnant chaque élément du réseau, il est possible d'en éditer les propriétés, comme, par exemple, le nombre de voies de circulation ou leurs largeurs pour les routes.

La carte ainsi créée est un arrangement planaire de courbes dans le plan. Un tel arrangement est une subdivision du plan en en cellules de dimensions 0, 1 et 2, respectivement appelées sommets, arêtes et facettes, et correspondant, du point de vue du réseau urbain, aux intersections, voies et îlots respectivement.

Pour le moment, les courbes utilisées ne peuvent être que des segments de droite. Néanmoins, grâce à l'utilisation d'algorithmes génériques, nous pourrons ajouter des segments de coniques par exemple.

Géométrie

La représentation topologique du réseau urbain nous permet une création simple de celui-ci. Le niveau d'abstraction suivant consiste à attribuer une géométrie à ces éléments topologiques. Ce niveau géométrique est généré automatiquement à partir des paramètres affectés aux éléments topologiques, en particulier la largeur attribuée aux voies. Ainsi, nous calculons les empreintes au sol de chaque élément du niveau topologique et la lui attribuons comme représentation géométrique. Cette dernière peut ensuite être elle-même décomposée. Nous allons maintenant traiter de telles décomposition pour le cas des îlots qui est le plus complexe.

5.3.3 Îlot et parcelle

Comme nous l'avons vu, l'étape de modélisation précédente nous fournit les informations nécessaires à pouvoir décomposer les éléments du réseau urbain.

En ce qui concerne la décomposition des îlots, nous proposons deux types de subdivision : la première décompose un îlots en îlots plus petits, la seconde en parcelles. Néanmoins, ces deux types de décomposition partagent de nombreux points communs et nécessitent des traitements similaires. Sans contrainte particulière sur les propriétés géométriques des îlots, il s'agit, dans les deux cas, d'un problème de subdivision de polygone. Nous considérons tout de même une contrainte sur ces polygones : ils doivent être *simples*. Cette contrainte n'est pas limitative sur la variété des polygones traités puisqu'un polygone est simple si ses arêtes ne s'intersectent qu'en leurs sommets[4].

Il existe un certain nombre de travaux sur la subdivision de polygones mais aucun n'intègre les contraintes propres à la subdivision d'îlots. En effet, la façon dont les îlots et les parcelles sont formés, leurs formes et leurs surfaces dépendent de phénomènes socio-culturels : de l'économie et de l'utopie, d'aménagements urbains globaux et de processus locaux. Nous trouvons, par exemple, un algorithme dichotomique de décomposition des îlots proposé par Parish et Müller [PM01]. Néanmoins,

[4]Les polygones simples sont parfois aussi appelés polygones ou courbes de Jordan du fait qu'elles vérifient le théorème du même nom. Ce théorème établit l'existence que les courbes de Jordan sont les seules à diviser le plan en deux composantes distinctes : l'intérieur du polygone et l'extérieur, son complément.

aucun détail n'est donné concernant cet algorithme et les cas complexes tels que les îlots concaves ne sont pas traités.

Au final, les îlots sont décomposés en parcelles constructibles. Une propriété essentielle de ces parcelles est ainsi l'accessibilité : pour que les bâtiments construits sur une parcelle soient accessibles, cette parcelle doit disposer d'un accès au réseau urbain. Habraken [Hab00] définit la profondeur territoriale comme le nombre de frontières de territoire à franchir pour accéder à un bâtiment ou un appartement. Il constate ainsi que, dans la plupart des sociétés dites occidentales, cette profondeur est faible, c'est à dire que la plupart des bâtiments disposent d'un accès direct à la rue. Dans certaines villes du moyen orient, comme Tunis, cette profondeur est plus élevée et certaines maisons ne peuvent être atteintes qu'en passant par des allées elles-mêmes privées, voire par d'autres maisons. Ainsi, en excluant le droit de passage, l'accès d'une parcelle à la rue est en effet une propriété essentielle. C'est pourquoi c'est tout d'abord cette propriété qui va nous intéresser dans la décomposition des îlots en parcelles.

Tout d'abord, afin de traiter la décomposition de polygones quelconques, il est nécessaire d'en construire une représentation structurée. Dans le contexte de la géométrie algébrique, trois outils principaux sont utilisés : le *diagramme de Voronoï*), l'*axe médian* et le *squelette droit* (straight skeleton).

5.3.3.1 Subdivision et squelettes

Considérons un ensemble E de sites dans l'espace Euclidien de dimension n où chaque site est un sous ensemble de cet espace :un point ou un segment de droite. On peut alors donner la définition suivante [Aur91, FEC02] :

Définition 11 (Région et Diagramme de Voronoï) *La* région de Voronoï *(ou cellule) d'un site e de E, dénoté $V_R(e)$, est l'ensemble des points plus près de e que de tout autre site de E :*

$$V_R(e) = \{p \in \mathbb{R}^n | d(p,e) \leq d(p,e'), \forall e' \neq e, e' \in E\}$$

où d est la distance Euclidienne. Le diagramme de Voronoï *de E, noté $VD(E)$, est défini comme l'union des frontières des régions de Voronoï :*

$$VD(E) = \bigcup_i \delta V_R(e_i)$$

où δ dénote la frontière d'un ensemble.

La figure 5.6 (a) illustre le diagramme de Voronoï (en bleu) d'un ensemble de sites (en rouge) définissant un polygone simple.

Considérons maintenant un sous-ensemble S de l'espace Euclidien de dimension n et sa frontière B.

Définition 12 (Axe médian) *Une boule ouverte à l'intérieur de B incluse dans aucune autre boule ouverte à l'intérieur de B est appelée* boule maximale. *L'*axe médian *de S est défini comme le lieu des centres de ses boules maximales*[5].

[5]On note que cette définition n'est applicable que sur des frontières décrivant un polygone simple, c'est à dire possédant un intérieur et un extérieur.

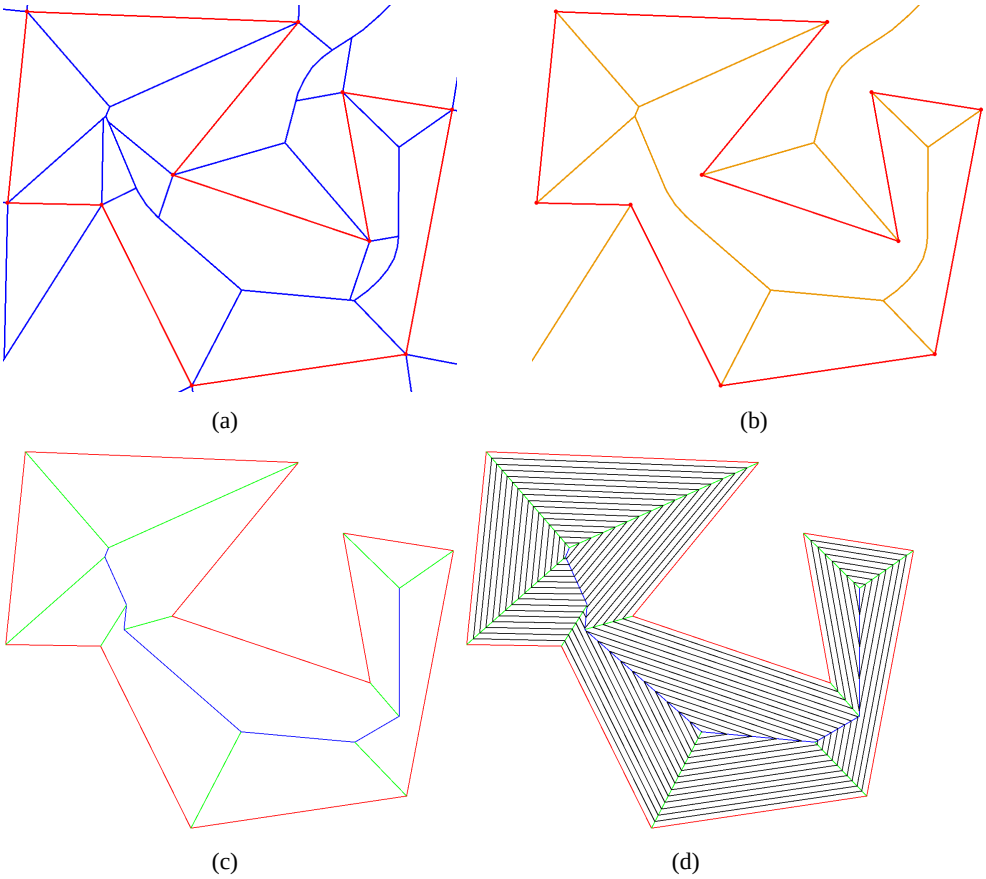


FIG. 5.6 – Différents types de squelettes pour un polygone simple (en rouge). (a) : le diagramme de Voronoï. (b) : l'axe médian. (c) : le squelette droit. (d) : polygones obtenus par le rétrécissement du polygone rouge (ou offset polygons) et définissant le squelettre droit illustré en (c).

La figure 5.6 (b) illustre l'axe médian (en bleu) d'un sous-ensemble de l'espace Euclidien $\mathbb{R}^2$ entouré d'une frontière (en rouge) définissant un polygone simple. Comme on peut le voir dans la figure, dans le cas de polygones simples, l'axe médian est un sous-ensemble du diagramme de Voronoï associé [FEC02] dans lequel on supprime les arêtes connectées à un sommet concave.

Définition 13 (Squelette droit) *Le rétrécissement d'un polygone simple P en utilisant une distance Euclidienne $t > 0$ produit un ensemble de polygones (offset polygons) par le décalage parallèle de chaque arête de P vers l'intérieur à une distance t. Le squelette droit $S(P)$ du polygone simple P est l'union des segments des bissectrices tracées par les sommets de P pendant un tel processus de rétrécissement.*

Le squelette droit (cf. figure 5.6 (c)) découpe ainsi tout n-gone simple P de façon unique en un ensemble de n polygones monotones : un pour chaque arête de P [AAAG95]. Il constitue un moyen simple de construire un toit polygonal sur un mur constitué par un polygone. Il permet aussi de définir une hiérarchie de polygones créés par décalage (cf. figure 5.6 (d)).

Le diagramme de Voronoï, l'axe médian et le squelette droit nous fournissent ainsi trois types de squelette pour représenter la structure d'un polygone simple. Le

squelette droit offre comme avantage une subdivision directe d'un polygone simple quelconque en polygones monotones. Les diagrammes de Voronoï, quant à eux, permettent d'obtenir des polygones (les régions) monotones sauf pour ceux qui correspondent aux sites concaves. Néanmoins, ceci nous permet de traiter ces régions indépendamment, ce qui peut s'avérer utile sachant que ces régions peuvent poser des problèmes d'accessibilité.

5.3.3.2 Décomposition des îlots en parcelles

Afin de pouvoir y construire des bâtiments accessibles par des voies, les îlots sont décomposés en parcelles. Ces parcelles possèdent par ailleurs des caractéristiques culturelles, historiques et fonctionnelles. En effet, les parcelles résidentielles sont, par exemple, généralement plus petites que les parcelles des grands ensembles. Il est donc important, afin de proposer des algorithmes de décomposition des îlots en parcelles, de déterminer des critères concernant la taille et la forme des parcelles. Dans le contexte urbain, nous proposons l'utilisation de critères simples tels que la surface, la largeur et la profondeur. Ainsi, pour un îlot donné, nous utilisons :

- la surface moyenne, minimale et maximale de chaque parcelle,
- la largeur moyenne, minimale et maximale de chaque parcelle,
- la profondeur moyenne, minimale et maximale de chaque parcelle.

Le problème des sommets concaves

L'utilisation d'un squelette nous permet de décomposer le contour d'un îlot en polygones monotones. Quelque soit le type de squelette utilisé, il se pose néanmoins un problème d'accessibilité au niveau des sommets concaves du contour. En effet, dans le cas de l'utilisation d'un squelette droit, une surface potentiellement importante des parcelles concernées n'est pas accessible. L'utilisation de diagrammes de Voronoï, quant à elle, permet de traiter ces zones indépendamment, et permet de garantir des propriétés intéressantes sur les polygones obtenus. Néanmoins, le problème de l'accès à ces zones reste, et nous proposons plusieurs solutions.

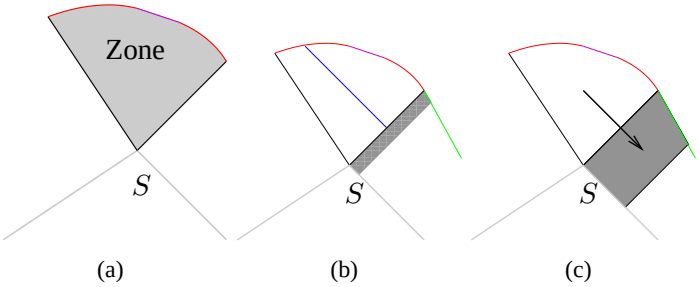


FIG. 5.7 – Traitement d'une zone concave correspondant au site S. (a) : Zone concave et son site. Les sites adjacents sont illustrés en gris clair. Les contours de la zone en noir, rouge et violet. (b) : Ajout d'une allée (en gris) pour donner accès à la zone concave. (c) : Attribution de la zone concave à une parcelle adjacente.

La première de ces solutions est de construire des parcelles dépendantes, c'est à dire de plus grande profondeur territoriale, dont l'accès dépend de l'accès aux parcelles adjacentes. Ceci correspond, dans une situation réelle, à un droit de passage. L'avantage de cette solution est bien sure la simplicité.

La seconde solution consiste à introduire une nouvelle voie offrant l'accès à ces zones. Soit au niveau du sommet concave (ce qui offre une solution proche du squelette droit), ce qui impose de modifier les deux parcelles adjacentes, soit directement au niveau de l'une de ces dernières.

Pour finir, la troisième solution consiste à attribuer la zone en question à une parcelle adjacente, de la découper pour l'attribuer à plusieurs, ou d'en faire un espace public. Ces différentes solutions sont illustrées par la figure 5.7.

Terminologie et structure de données

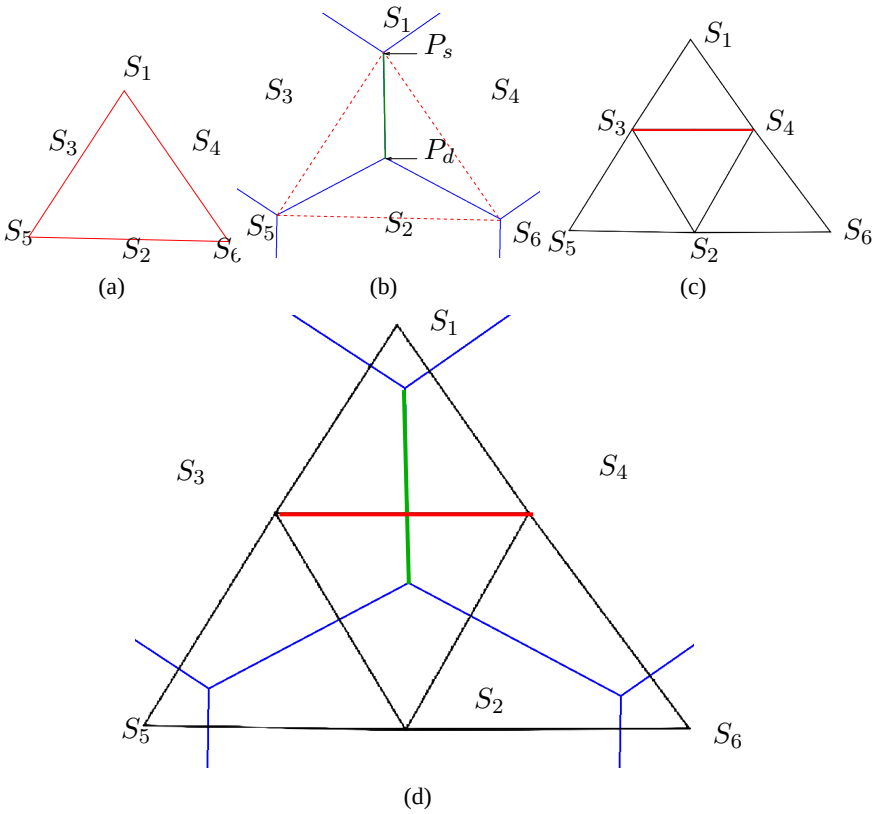


FIG. 5.8 – Relations entre un polygone simple (en rouge), son diagramme de Voronoï (en bleu) et son dual (en noir). (a) : le polygone traité. (b) : son diagramme de Voronoï. Les sites S_1, S_2, S_3 et S_4 définissent l'arête d'extrémités P_s et P_d (en vert). (c) : le dual du diagramme de Voronoï. Les sommets du dual sont associés aux cellules (et donc aux sites) du diagramme de Voronoï. Chaque arête du dual est associée à une arête du diagramme de Voronoï, ainsi, l'arête reliant les sites S_3 et S_4 (en rouge) correspond à l'arête du diagramme de Voronoï séparant les sites S_3 et S_4 (en rouge). (d) : le diagramme de Voronoï et son dual superposés.

Pour ce type de décomposition, nous utilisons des diagrammes de Voronoï. Chaque site du diagramme nous donne une surface que nous pouvons alors décomposer en parcelles. Nous allons tout d'abord définir une terminologie sur les arêtes du diagramme que nous allons utiliser. Chaque arête du diagramme est définie par quatre sites : les deux sites (S_3 et S_4) dont elle sépare les cellules (dont elle est équidistante), et les deux sites (S_1 et S_2) qui définissent ses extrémités (P_s et P_d), comme illustré par la figure 5.8.

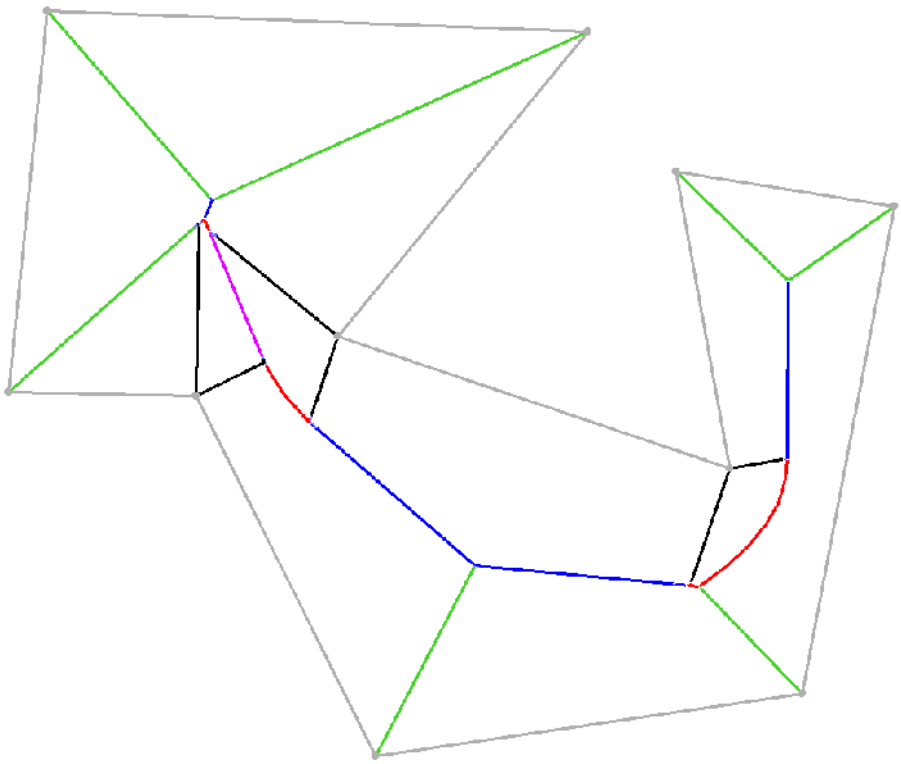


FIG. 5.9 – Différents types d'arêtes sur le squelette d'un diagramme de Voronoï. En gris : le polygone supportant les sites du diagramme. En vert : les arêtes secondaires. En bleu : les arêtes primaires. En rouge : les arêtes paraboliques. En noir et en violet : les arêtes dites concaves. L'arête en violet définit la frontière des cellules de deux sites ponctuels concaves.

Les différents types d'arêtes utilisées sont donc les suivants (voir figure 5.9) :

- *Arête parabolique* : une arête est dite parabolique si elle définit la frontière des cellules d'un site ponctuel et d'un segment (l'arête décrit alors un segment de parabole).
- *Arête concave* : une arête est dite concave si l'un de ses sommets est un site ponctuel concave ou si elle définit la frontière des cellules de deux sites ponctuels concaves.
- *Arête secondaire* : une arête est dite secondaire si l'un de ses sommets appartient au diagramme (au polygone initial).
- *Arête primaire* : une arête est primaire si elle n'est ni concave ni secondaire, i.e. si elle n'a pas de sommet sur le polygone initial.

Nous définissons ainsi la structure de données des arêtes et des sites. Chaque arête A possède :

- deux sommets, ses extrémités, nommés P_s et P_d (source et destination).
- quatre sites. Deux qui déterminent les extrémités de l'arête S_1 et S_2. Deux dont l'arête sépare les cellules S_3 et S_4.

De même, chaque site S possède :

- un type : point ou segment.

Marquer toutes les arêtes du diagramme de Voronoï du polygone $Poly$ comme primaire, secondaire, parabolique ou concave ;
Créer la liste des lotissements L_{lot} ;
Créer la liste des parcelles L_{par} ;
Pour chaque arête secondaire A {
    Si $Poly.isOnBoundary(A.P_s)$ alors {$P_i = A.P_d$; $P_j = A.P_s$; }
    Sinon { $P_i = A.P_s$; $P_j = A.P_d$; }
    $P_3 = projection(P_i, A.S_3.segment())$;
    $P_4 = projection(P_i, A.S_4.segment())$;
    $Aire_{angle} = Polygon(P_j, P_3, P_i, P_4).area()$;
    Si $Aire_{angle} < Aire_{max}$ alors $L_{par}.insert(Polygon(P_j, P_3, P_i, P_4))$;
    Sinon {
        Placer un point P sur l'arête A ;
        $P'_3 = projection(P, A.S_3.segment())$;
        $P'_4 = projection(P, A.S_4.segment())$;
        $L_{par}.insert(Polygon(P_j, P'_3, P, P'_4))$;
        $L_{lot}.insert(Polygon(P'_3, P_3, P_i, P))$;
        $L_{lot}.insert(Polygon(P_4, P'_4, P, P_i))$;
    }
}
Pour chaque arête primaire A {
    $P_i3 = projection(P_i, A.S_3.segment())$;
    $P_i4 = projection(P_i, A.S_4.segment())$;
    $P_{j3} = projection(P_j, A.S_3.segment())$;
    $P_{j4} = projection(P_j, A.S_4.segment())$;
    $L_{lot}.insert(Polygon(P_{i3}, P_{i4}, P_j, P_i))$;
    $L_{lot}.insert(Polygon(P_i, P_j, P_{j4}, P_{j3}))$;
}
Pour chaque arête parabolique A {
    Si $A.S_3.isSegment()$ alors $S = S_3$;
    Sinon $S = S_4$;
    $P'_i = projection(P_i, S.segment())$;
    $P'_j = projection(P_j, S.segment())$;
    $L_{lot}.insert(Polygon(P'_i, P'_j, P_j, P_i))$;
}
Pour chaque lotissement L de L_{lot} {
    Si $L.area() < Aire_{max}$ alors $L_{par}.insert(L)$;
    Sinon {
        Si $Segment(L.P_1, L.P_2).lenth() < Length_{max}$ alors {
            $P'_1 = \frac{P_1+P_4}{2}$;
            $P'_2 = projection(P'_1, Segment(P_2, P_3))$;
            $L_{lot}.insert(Polygon(P_1, P_2, P'_2, P'_1))$;
            $L_{lot}.insert(Polygon(P'_1, P'_2, P_3, P_4))$;
        }
        Sinon {
            $L_{lot}.insert(Polygon(P_1, \frac{P_1+P_2}{2}, \frac{P_3+P_4}{2}, P_4))$;
            $L_{lot}.insert(Polygon(\frac{P_1+P_2}{2}, P_2, P_3, \frac{P_3+P_4}{2}, P_4))$;
        }
    }
}

FIG. 5.10 – Algorithme de décomposition des îlots en parcelles.

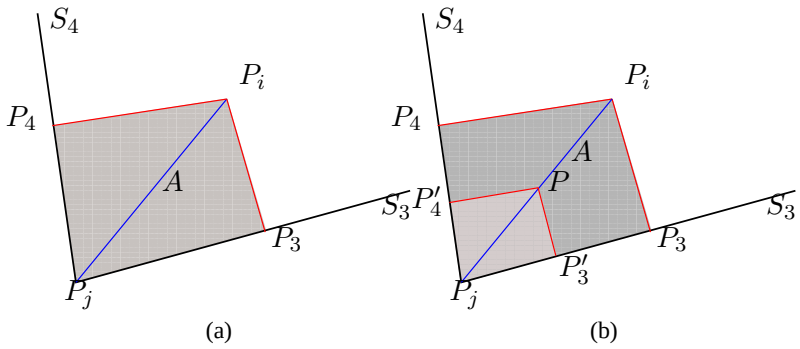


FIG. 5.11 – Illustration du découpage d'une arête secondaire A et de la création d'une parcelle d'angle. (a) Polygone obtenu après projection de P_i sur les sites adjacents S_3 et $S4$. (b) Polygones obtenus après découpage de l'arête A en un point P et la projection de ce dernier sur S_3 et $S4$. Les polygones formés par (P_3, P_i, P, P'_3) et (P_3, P_i, P, P'_3) et illustrés en gris plus foncé sont des lotissements. Ces derniers sont traités séparément.

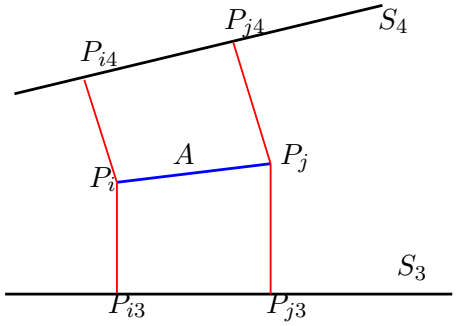


FIG. 5.12 – Illustration du traitement d'une arête primaire A. Les points P_i et P_j sont projetés orthogonalement sur les sites S_3 et S_4. Les deux polygones résultant sont traités comme des lotissements.

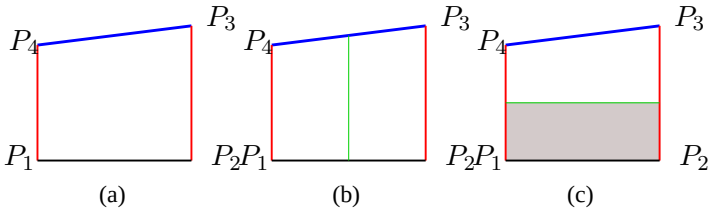
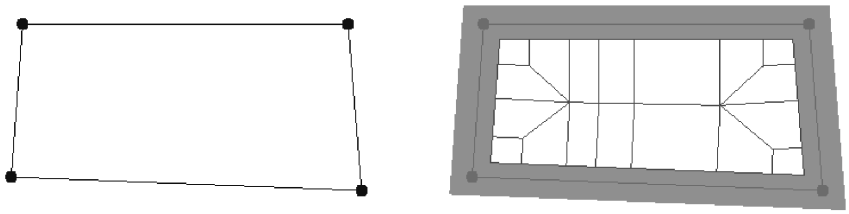
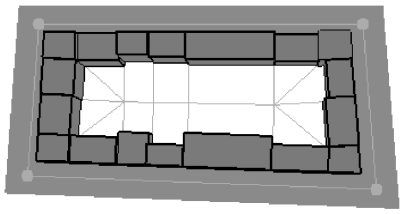


FIG. 5.13 – Découpage d'un lotissement en parcelles. (a) : Lotissement initial. L'arête marquée en noir représente l'accès du lotissement à la route, les arêtes rouges lui sont perpendiculaires, et l'arête bleue représente le fond du lotissement. (b) : Découpage longitudinal du lotissement, c'est à dire le long de l'accès au réseau routier. (c) : Découpage en profondeur créant un niveau de profondeur territoriale supplémentaire. La parcelle résultante en gris est marquée comme traversante, c'est à dire qu'elle doit permettre l'accès à la parcelle ségrégée.

- deux sommets, ses extrémités, nommés P_s et P_d (source et destination) si le site est un segment.
- un seul sommet, ses extrémités, nommé P si le site est un point.



(a) (b)



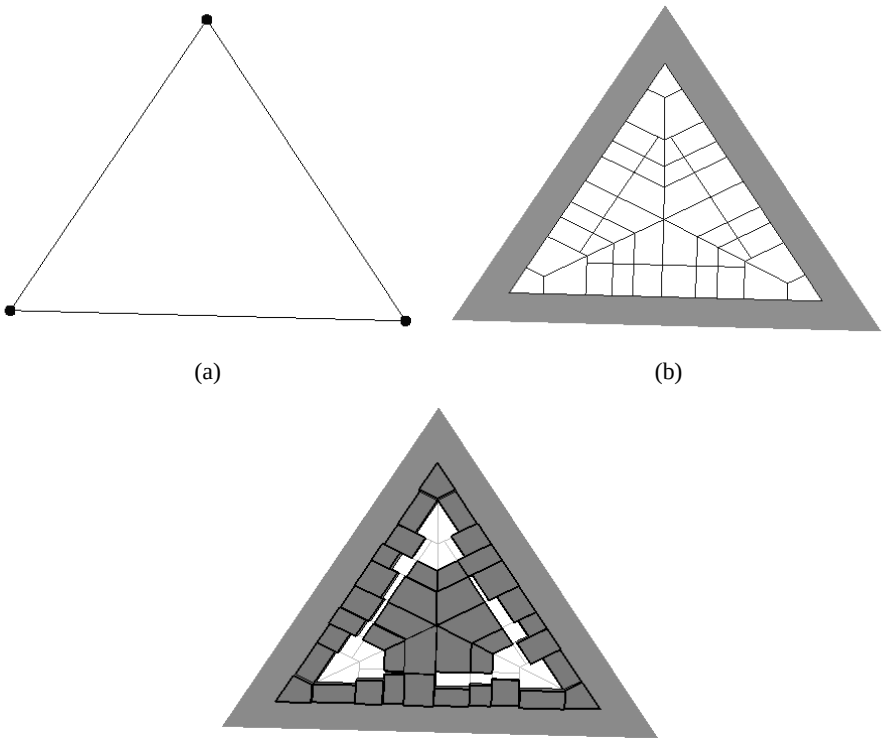
(c)

FIG. 5.14 – Construction des parcelles d'un îlot Simple. (a) : tracé des routes entourant l'îlot. (b) : empreintes des routes et des parcelles. (c) : placement d'enveloppes de bâtiments.

L'algorithme présenté en figure 5.10 décompose le polygone $Poly$ en polygones simples appelés parcelles. Les lotissements sont des polygones simples intermédiaires qui sont eux-mêmes décomposés en parcelles. La fonction $Poly.isOnBoundary(P)$ permet de tester si le point P est sur le contour du polygone $Poly$. La fonction $projection(P, S)$ projette le point P orthogonalement sur le segment S et donne un point comme résultat. Ici, nous projetons le point P_i (cf. figure 5.10) sur les sites S_3 et S_4. Ici, tester si les sites S_3 et S_4 sont bien des segments est inutile puisque c'est une condition pour que l'arête A soit secondaire. La figure 5.11 illustre le traitement des parcelles d'angle, la figure 5.12 celui des arêtes primaires et la figure 5.13 le traitement des lotissements. L'algorithme donné ici ne traite pas les zones dites concaves dont nous avons parlé précédemment. L'un des avantages de notre algorithme est d'ailleurs de pouvoir traiter ces derniers indépendamment. Les figures 5.14, 5.15 et 5.16 illustrent l'algorithme.

Marquage des arêtes et polygones

Notre algorithme marque, au fur et à mesure de son exécution, les arêtes et polygones générés. En particulier, les parcelles sont marquées par leur profondeur territoriale. De même, une parcelle traversante est marquée comme telle. De plus, chacune des arêtes des parcelles produites est marquée comme ayant accès ou non au réseau, comme séparant deux parcelles de même profondeur territoriale ou comme séparant des parcelles de profondeurs territoriales différentes : elle est alors traversante. Ces marqueurs sur les arêtes et les parcelles nous permet, notamment, de déterminer rapidement certaines propriétés des polygones comme la perpendicularité de certaines arêtes. Ceci est utilisé pour la création des empreintes au sol des bâtiments, et pourrait être utilisé, par exemple, pour placer des porches sous les bâtiments traversants,



(a) (b)

(c)

FIG. 5.15 – Construction des parcelles d'un îlot triangulaire. (a) : tracé des routes entourant l'îlot. (b) : empreintes des routes et des parcelles. (c) : placement d'enveloppes de bâtiments. On notera que l'îlot présente une profondeur territoriale de deux niveaux alors que celui illustré en figure 5.14 n'en présente qu'un seul.

pour ajouter des allées, etc.

5.3.3.3 Subdivision des îlots

La subdivision des îlots répond à deux nécessités. D'une part, elle reflète un des processus liés aux grands aménagements urbains qui procèdent en premier lieu par un découpage des aménagements en zones ou quartiers, puis, en second lieu, de ces zones en îlots. D'autre part, il est utile à la modélisation procédurale afin de découper les îlots dont les caractéristiques ne répondent pas à certains critères. En effet, certains îlots, modélisés par l'utilisateur ou générés, peuvent être, par exemple, trop vastes ou trop profonds. Nous proposons ainsi quelques critères pouvant être utilisés pour déterminer quels îlots doivent être subdivisés, ainsi qu'un algorithme pour effectuer la subdivision.

Nous traitons en particulier ici des problèmes d'accessibilité : en effet, les îlots sont une enclave à l'intérieure de laquelle sont tracées des parcelles afin d'y construire des bâtiments. Ces derniers doivent être accessibles, ce qui impose des contraintes sur le placement des bâtiments, mais aussi sur la forme des îlots. Par exemple, un îlot habité d'un kilomètre sur un kilomètre en centre ville est inconcevable à moins d'être parcouru par un réseau de voies permettant d'accéder aux bâtiments les plus isolés, c'est à dire dont la profondeur territoriale est la plus grande. Ceci nous permet de poser un critère simple utilisant la distance entre le squelette du contour de l'îlot

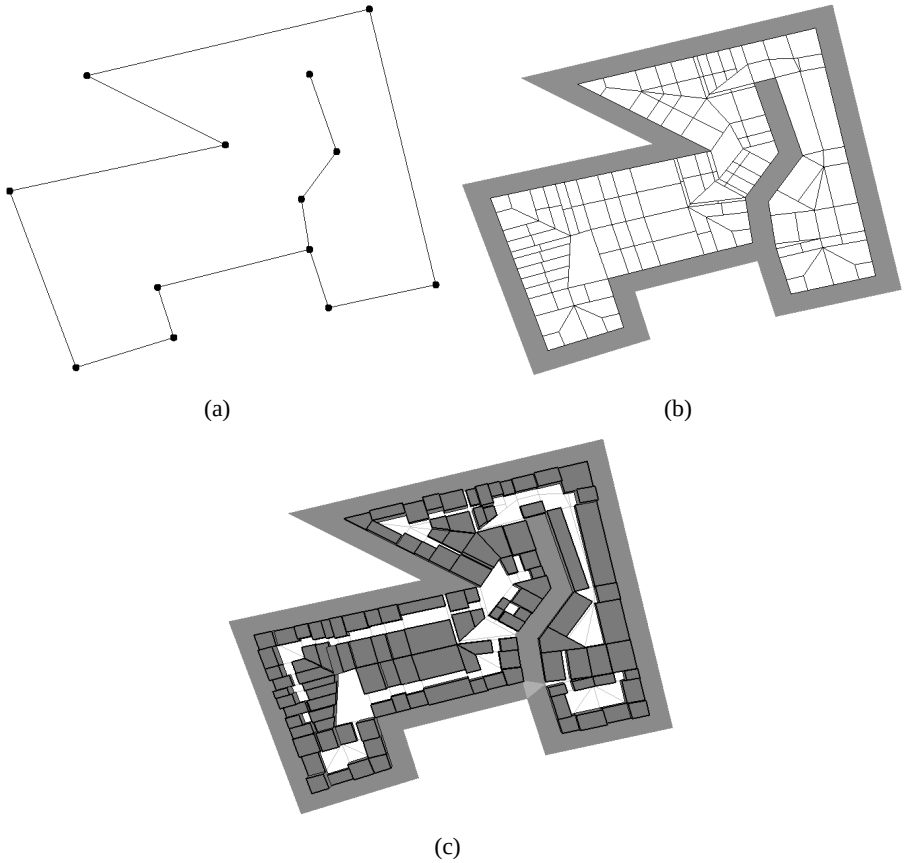


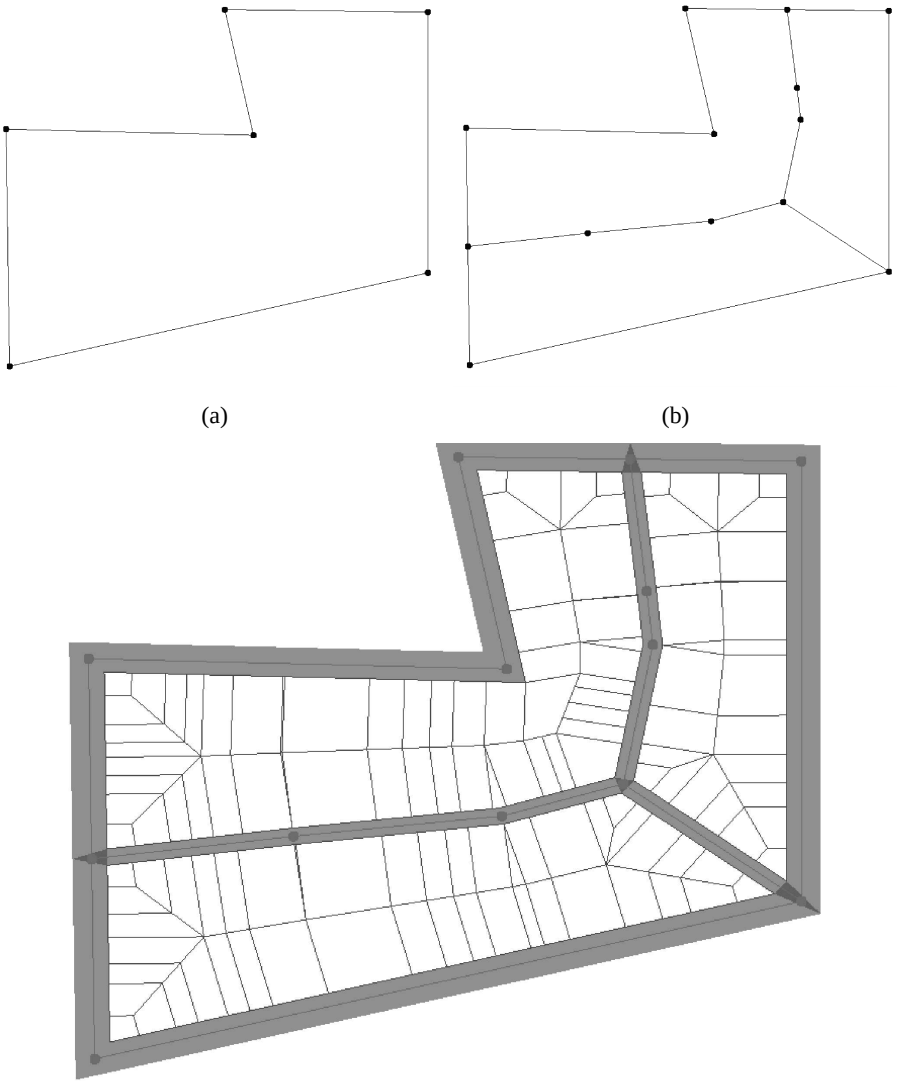
FIG. 5.16 – Construction des parcelles d'un îlot complexe. (a) : tracé des routes entourant l'îlot. (b) : empreintes des routes et des parcelles. (c) : placement d'enveloppes de bâtiments. On notera la présence, ici d'une impasse à l'intérieur de l'îlot.

et le contour lui-même : la profondeur maximale. Ainsi, si la profondeur maximale d'un îlot est atteinte, celui-ci est subdivise en îlots plus petits.

L'algorithme consiste à ajouter au réseau urbain certaines arêtes du dual du diagramme de Voronoï de l'îlot. Toutes les arêtes primaires sont ajoutées. En ce qui concerne les arêtes secondaires, elles sont ajoutées, à l'exception de celles qui sont connectées à un site lui-même connecté à deux arêtes secondaires : dans ce cas, une arête est ajoutée entre le sommet intérieur de ces deux arêtes et le milieu du site concerne. La figure 5.17 illustre l'algorithme.

5.3.4 cadastre

À partir des parcelles générées et des paramètres définis pour le quartier, le cadastre est généré. Nous ne proposons ici qu'un seul algorithme simple de placement des bâtiments au ras de la rue ou au fond de la parcelle. Nous profitons ici d'une propriété importante des parcelles générées : les arêtes situées de chaque cote de la façade lui sont perpendiculaires, à l'exception des parcelles d'angles et des parcelles associées aux sommets concaves des îlots (dites parcelles concaves par abus de langage). Pour les parcelles d'angle, nous générons une empreinte de bâtiment dont l'empreinte au sol est la parcelle entière. Pour les parcelles dits concaves, nous ne les traitons pas pour l'instant : elles sont en fait affectées à l'une des parcelles adjacente



(a) (b)

(c)

FIG. 5.17 – Subdivision d'un îlot trop profond en îlots plus petits. (a) : Îlot initial. (b) : Arêtes du diagramme de Voronoï retenues et ajoutées au réseau urbain. On remarque que les arêtes reliées au sommet concave ont été supprimées. De plus, les arêtes secondaires qui auraient dû relier quatre des sommets de l'îlot initial ont été remplacées par des arêtes coupant les arêtes de l'îlot initial en deux. (c) : Résultat après décomposition des îlots en parcelles.

comme dépendance (jardin ou parking par exemple).

Pour les autres, la configuration est simple : nous avons une arête représentant le devant de la parcelle, deux arêtes perpendiculaires à la première (et donc parallèles entre elles), représentant les côtés de la parcelle, et une dernière arête, le fond de parcelle. Ainsi, pour créer une empreinte au sol de bâtiment simple, nous traçons une droite parallèle à la façade (c'est à dire au devant de parcelle) et décalée de la profondeur voulue (déterminée par les paramètres de l'îlot ou du quartier). Les intersections de cette droite avec les côtés de la parcelle nous donnent les extrémités du fond de l'empreinte au sol du bâtiment.

En ce qui concerne le calcul de la profondeur du bâtiment, nous tentons de respecter les profondeurs, largeurs et surfaces indiquées par les paramètres de l'îlot. Si l'une de ces propriétés ne peut être respectée, aucun bâtiment n'est placé et la parcelle est marquée comme espace public (parc), parking ou autre.

Pour finir, la hauteur des bâtiments est déterminée par une carte fournie par l'utilisateur ou, à défaut de carte, aléatoirement. Les figures 5.14, 5.15 et 5.16 présentent des exemples de placements d'enveloppes simplifiées de bâtiments, et la figure 5.18 illustre les cas particuliers de la construction des cadastres à l'exception des parcelles d'angle.

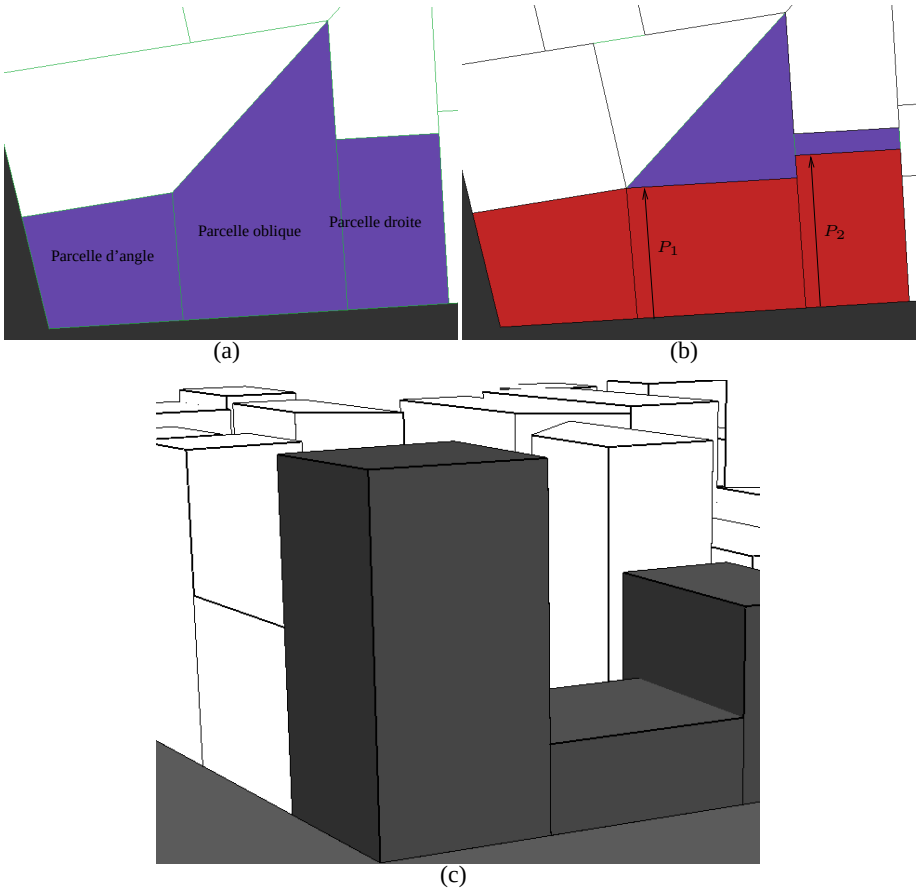


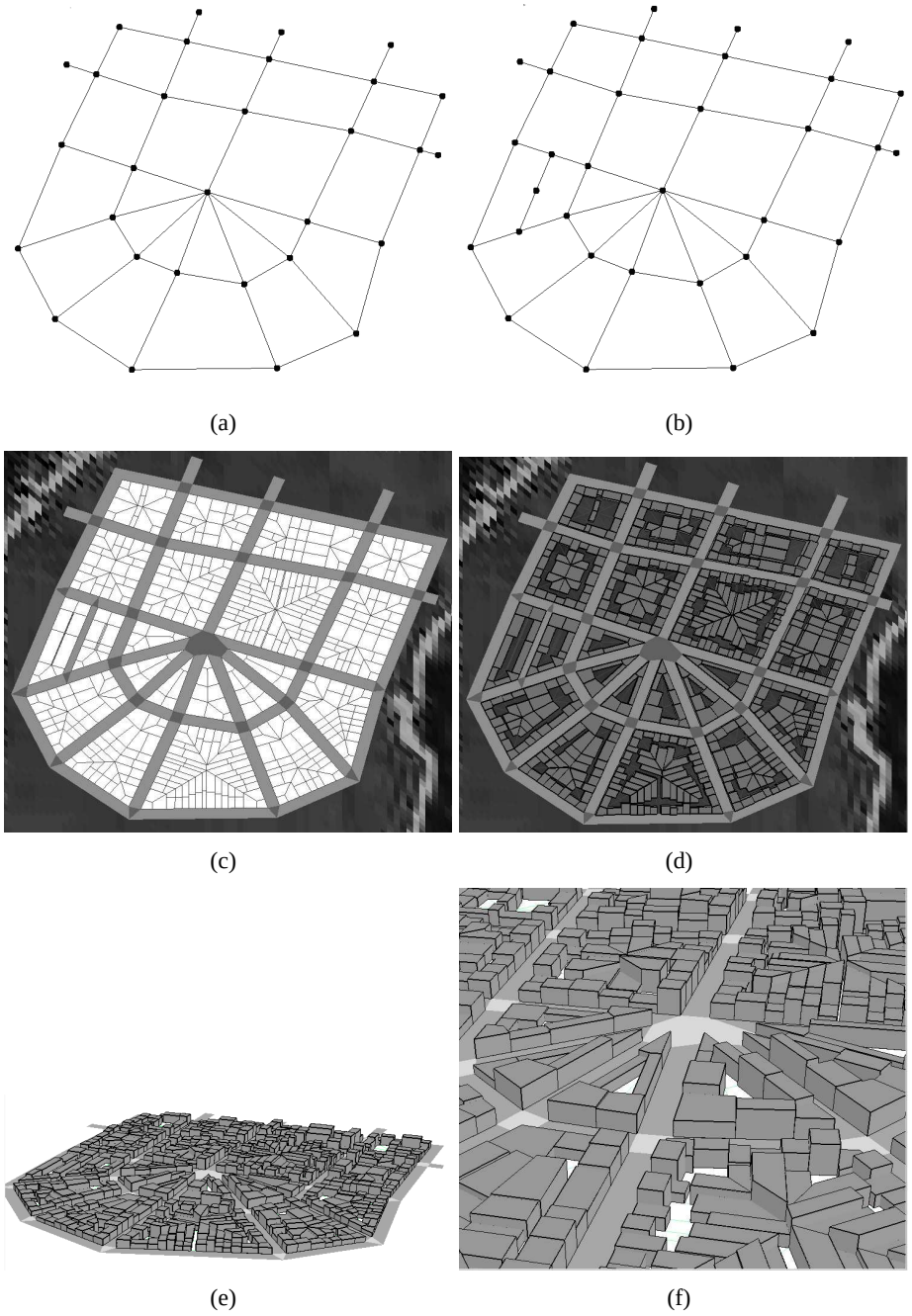
FIG. 5.18 – Cas particuliers de cadastres. (a) : Les trois parcelles considérées, une parcelle d'angle, une parcelle à fond oblique et une parcelle droite, c'est à dire un parallélogramme. (b) : Cadastre résultat avec, en rouge, l'empreinte du bâtiment et, en violet, la surface de parcelle restante (jardin, cour). (c) : Les enveloppes des bâtiments correspondant.

5.4 Résultats

Nous présentons ici des exemples d'utilisation du modèle proposé présentés dans les figures 5.19 et 5.20. Pour chacun des exemples présentés, la technique de modélisation est la même. L'utilisateur commence par charger une carte d'élévation (ici, la carte illustrée par la figure 5.5) qui permet de créer le modèle du terrain. Ensuite, l'utilisateur met en place les grands axes de la ville modélisée ainsi que sa forme gé-

nérale, sa macrofrome (cf. figures 5.19(a) et 5.20(a)). Le logiciel subdivise alors les îlots puis les décomposent en parcelles (cf. figures 5.19(b) et 5.20(b)).

Il faut noter que dans ces deux exemples, les informations fournies par l'utilisateur sont minimes : une carte d'élévation ainsi que le tracé général du modèle (première image de chaque figure). Le reste est généré automatiquement en quelques dizaines de secondes.



(a) (b)

(c) (d)

(e) (f)

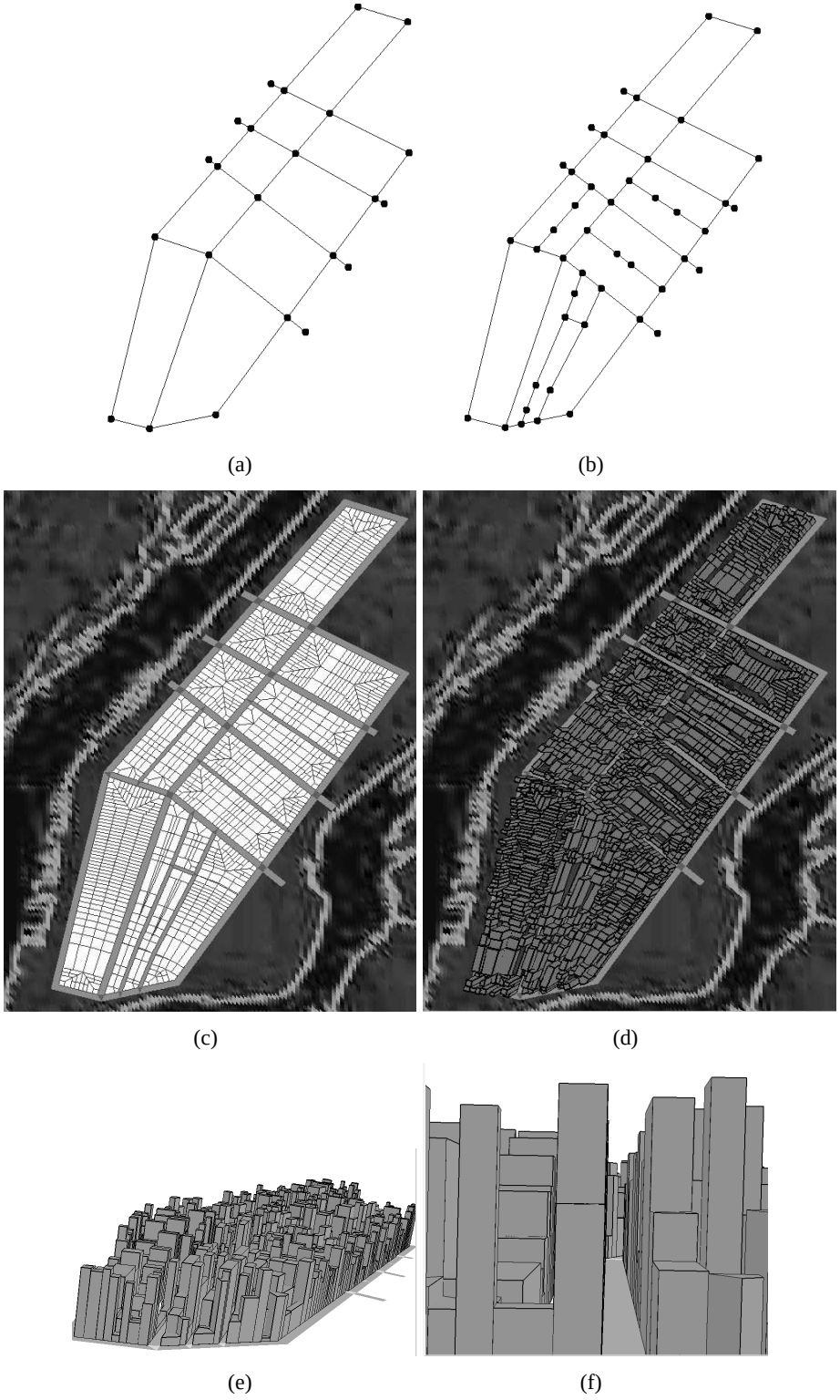
FIG. 5.19 – Différentes vues d'un premier modèle de ville. Sur la première ligne : le graphe urbain fourni par l'utilisateur, puis subdivisé automatiquement. Sur la seconde ligne : le parcellaire et le cadastre, tous deux générés automatiquement. Sur la dernière ligne : des vues aériennes du modèle. Dans ce modèle, les hauteurs des enveloppes de bâtiments sont produites aléatoirement entre 4 et 8 étages.

5.5 Conclusion et discussion

Dans ce chapitre, nous avons présenté une structure de données hiérarchique originale pour représenter les environnements urbains, ainsi que des algorithmes permettant de l'exploiter. Cette structure de données facilite grandement le processus de modélisation d'environnements urbains, qu'il soit effectué par un utilisateur ou entièrement généré. À cette fin, nous avons proposé plusieurs contributions. Tout d'abord, nous avons présenté des algorithmes de décomposition et de subdivision des îlots travaillant sur des polygones simples quelconques. Ces derniers permettent de découper les îlots en respectant des propriétés données facilement extensibles. Par ailleurs, nous avons proposé une hiérarchie d'abstraction pour les environnements urbains permettant de structurer ces derniers mais aussi, intégrés à une plate-forme de modélisation, autorisent l'utilisateur à se concentrer sur les informations qui lui sont utiles. Cette technique améliore de plus les performances de la visualisation et permet, une modélisation interactive et intelligente de la ville.

La structure de données présentée est finalement complémentaire à des techniques procédurales plus spécialisées comme celles présentées aux chapitres précédents. Et il est prévu de combler le vide qui sépare pour l'instant ce modèle et les *FL-systems* par exemple. Encore une fois, les hiérarchies d'abstraction s'avèrent très intéressantes. Elles pourront, par exemple, nous aider à mettre en place un mécanisme d'évaluation paresseuse sur les objets procéduraux du modèle. Ainsi, l'utilisateur pourra, lors de la modélisation d'un objet aussi complexe qu'une ville, s'intéresser à une place en particulier, à un bâtiment, à un arbre…

Ces travaux permettront par la suite de mettre en place une étude de la croissance de la ville, et notamment de l'évolution et de la propagation des attributs morphologiques propres à une période historique, mais aussi de la démographie, des réseaux sociaux, de l'utilisation du sol et du renouvellement urbain. En effet, nous l'avons vu dans le cas des *FL-systems*, les méthodes procédurales sont plus efficaces et simple à utiliser losque l'on sépare les processus différents qu'elles modélisent. Et il en est de même pour les villes que pour les plantes : il est important de séparer les processus de décomposition et de croissance. Les processus de décomposition, que nous avons partiellement traités dans ce chapitre, consistent à modéliser la structuration, à un instant donné, de la ville. En revanche, les processus de croissance modélisent l'évolution dans le temps des éléments urbains. Cette étude, que nous avons commencée, est passionante mais très complexe de par la diversité des acteurs de ce changement, des forces à l'œuvre, et des utopies mises en place, etc. Parmi les autres perspectives, l'intégration du parcellaire agraire nous parait intéressante puisqu'il a une influence capitale sur la morphologie urbaine. Pour finir, pour compléter l'analogie avec le modèle de Lynch, il est nécessaire d'intégrer les notions de quartier, qui sont ici des agrégations d'îlots, et les points de repère.



(a) (b)

(c) (d)

(e) (f)

FIG. 5.20 – Différentes vues d'un autre modèle de ville. Sur la première ligne : le graphe urbain fourni par l'utilisateur, puis subdivisé automatiquement. Sur la seconde ligne : le parcellaire et le cadastre, tous deux générés automatiquement. Sur la dernière ligne : des vues aériennes du modèle. En ce qui concerne la hauteur des enveloppes de bâtiments, elles ont été ici fournies par l'utilisateur au moyen d'une carte d'élévation.

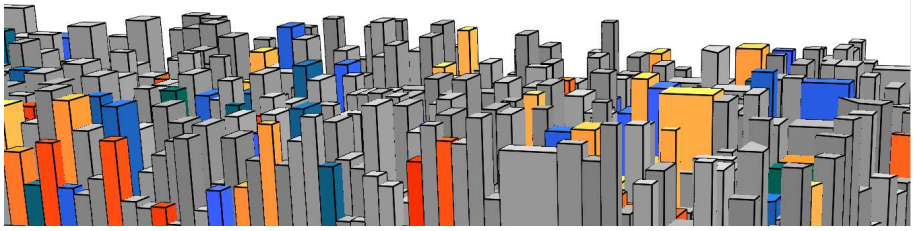


FIG. 5.21 – Vue d’un modèle virtuel de ville créé.

Chapitre 6

Conclusion et perspectives

6.1 Contributions

Dans cette thèse, nous avons présenté un état de l'art de la modélisation urbaine ainsi que deux nouveaux modèles pour représenter la ville. Les modèles proposés sont, à l'heure actuelle, utilisés pour une modélisation interactive d'environnements urbains imaginaires. Toutefois, ces derniers peuvent être créés à l'aide de données réelles comme les modèles numériques d'élévation afin de générer le terrain sur lequel est bâtie la ville virtuelle. Les techniques proposées pourront ainsi être utilisées pour la création interactive d'environnements urbains pour le cinéma (films d'animation, effets spéciaux), les jeux vidéos ou certains projets d'urbanisme dont la ressemblance du modèle avec la réalité importe peu, voire sans contexte préalable (conception de villes nouvelles par exemple). De plus, nous pensons que ces modèles, une fois généralisés à l'ensemble des éléments constituant l'espace urbain et spécialisés afin de pouvoir en représenter toute la diversité, pourront être utilisés pour modéliser plus fidèlement des environnements réels. Mais une telle utilisation, devant mettre en place des techniques complexes issues du domaine de l'analyse/synthèse, et faisant collaborer les méthodes de génération procédurale avec celles de l'analyse d'image, dépasse de loin la portée de cette thèse.

Dans ce document, nous avons montré, dans un premier temps, que l'utilisation de connaissances *a priori* sur les objets modélisés donne de meilleurs résultats. D'une part parce que ces connaissances permettent la mise en place de méthodes de modélisation adaptées et accélérées. D'autre part car elles incitent à la définition de représentations spécialisées dont l'utilisation améliore aussi bien les performances des applications que notre compréhension des objets étudiés. Notre proposition, à ce sujet, est que, pour simuler le fonctionnement du processus de modélisation traditionnellement réalisé par l'homme, il est intéressant de s'inspirer des processus et représentations du cerveau humain. Bien sûr, de grandes limitations existent à ce sujet du fait de notre manque de compréhension globale de tels processus et représentations, mais n'est-ce pas justement par des études trans-disciplinaires que de nouvelles perspectives peuvent naître ? Et pour comprendre le fonctionnement de notre cerveau, n'est-il pas primordial d'en étudier les codes, non pas seulement de façon abstraite ou empirique, mais de façon formelle, théorique et en se focalisant sur des tâches spécifiques ? Dans le cas bien particulier de la représentation de la ville, ces codes existent sur plusieurs niveaux, aussi bien structurel, à travers la morphologie urbaine, que sémiotique, composant des images complexes, mais aussi sociologique, grâce aux connections que les

réseaux sociaux créent et aux distances qu'ils réduisent.

C'est donc cette démarche que nous avons adoptée pour proposer un certain nombre de concepts et de méthodes décrits par les contributions suivantes :

- Une synthèse des différentes méthodes de modélisation de la ville en termes sémiotiques offrant une nouvelle perspectives sur le rapport de ces méthodes à l'objet modélisé et à l'utilisateur.
- La définition d'une extension fonctionnelle des *L-systems*, les *FL-systems* permettant la génération de la géométrie au fil de la réécriture.
- La définition d'un mécanisme de cache intelligent pour les *FL-systems*.
- L'application des *FL-systems* à la modélisation urbaine, à la simulation du processus de construction et à la description d'éléments architecturaux.
- La description d'une nouvelle structure de données hiérarchique pour représenter la ville basée sur les hiérarchies d'abstraction.
- La définition d'algorithmes pour exploiter cette structure de données et d'une architecture de modélisation.

Le domaine de la modélisation urbaine est vaste et s'étend, du moins dans la définition que nous en avons adoptée, de la modélisation détaillée des éléments architecturaux à celle des réseaux urbains dans toute leur complexité. Nous avons ainsi abordé ces deux extrêmes de la modélisation urbaine en utilisant des outils théoriques différents : les systèmes de réécriture d'un côté, la topologie et les hiérarchies d'abstraction de l'autre. Même si nous ne présentons pas ici d'application faisant le lien entre ces techniques, cette combinaison est à portée de main. Si nous avons utilisé des outils différents, c'est pour utiliser au mieux les avantages respectifs de ceux-ci vis à vis des objets modélisés. En effet, les motifs architecturaux se prêtent relativement bien à l'utilisation de langages pour les décrire mais conviennent difficilement à la description de réseaux urbains. À ce sujet, les systèmes de réécriture de graphes, par exemple, pourraient fournir des solutions potentielles. Réciproquement, les cartes topologiques représentent bien les réseaux urbains, mais ne présentent pas les mêmes qualités que les grammaires pour décrire les motifs architecturaux. Néanmoins, des recherches prometteuses existent quant à leur utilisation pour de telles applications [FML06].

Par delà la simple utopie, peut-être, d'une approche s'inspirant des processus cognitifs de l'homme dans la modélisation urbaine, il s'agit finalement de franchir cette barrière linguistique que décrivaient Barthes et Saussure. Vis à vis de cet objectif, le travail restant à effectuer est colossal et notre effort dans ce sens probablement négligeable. Mais de nombreux indices semblent aujourd'hui montrer que cette direction de recherche est une étape clé pour la création d'outils intelligents ou seulement adaptés à la modélisation urbaine. Il s'agit ainsi d'utiliser tous les outils à notre disposition pour comprendre, décrire et analyser la ville, pour finalement parler son langage sans utiliser de métaphore, pour paraphraser Barthes. Par ailleurs, cette approche est finalement dans la continuité des hypothèses posées par Dijkstra [Dij72] : les meilleurs langages sont spécifiques à une application et à un objet d'étude. Les outils de modélisation 3D prennent d'ailleurs timidement cette direction, et on voit apparaître de nombreuses extensions à ces logiciels pour l'architecture, la décoration intérieure, etc. Mais ces efforts se limitent, aujourd'hui, à des bibliothèques d'objets et de matériaux. On se limite ainsi à la seule dimension lexicale du langage de l'architecture et de la ville, en oubliant la syntaxique et la dimension sémantique voire sémiotique…

6.2 Perspectives

FL-systems

Nous somme actuellement en train de terminer une implémentation du mécanisme de gestion de cache pour les *FL-systems* proposé en section 4.4. Ce cache devra permettre de réduire significativement les temps de réécriture et confirmer l'avantage d'utiliser ces méthodes en temps réel, et non pas seulement hors-ligne pour générer des modèles complexes. En ce qui concerne les textures procédurales, la société *Algorithmic* [1] propose d'ailleurs un générateur de textures à la volée, *proFXengine*[2] dont le temps de génération est inférieur au temps de chargement du fichier bitmap. Nous pensons que notre système de cache permettra de se rapprocher d'un tel but pour la géométrie procédurale.

Malgré leur intérêt, nous pensons que l'utilisation des *FL-systems* pour la modélisation de bâtiments, par exemple, est grandement limitée par la complexité actuelle du langage. Ainsi, la création d'une interface graphique dédiée paraît nécessaire. À l'aide d'une telle interface, il deviendra simple et rapide de créer de nouveaux styles architecturaux en utilisant des éléments déjà définis. Ceci soulève aussi des questions quant à la création de bibliothèques de règles de réécriture que nous n'avons qu'effleurée dans cette thèse.

Pour finir concernant les *FL-systems*, une de nos arrière pensées pour l'utilisation de fonctions comme terminaux du langage concernait l'utilisation de fonctions du second degré dans les règles. En effet, grâce à l'utilisation du second degré, il serait, par exemple, possible de définir une règle de composition architecturale décomposant les éléments d'une façade et prenant comme paramètre une autre règle en charge du placement des pierres. Ceci permettrait la définition de règles plus génériques (des *templates*) pouvant être utilisées de façon plus large, et donc une meilleure réutilisation de celles-ci.

Modélisation

Nous avons montré la simplicité de modélisation qu'offre le dessin vectoriel combiné aux hiérarchies d'abstraction. Néanmoins, nous ne fournissons pas, pour le moment, les outils nécessaires à une modélisation plus fine de la géométrie des routes et des trottoirs par exemple. Des outils comme *VUEMS* [Don97, Tho99, Don04] présentent ce genre d'outils, et nous envisageons d'ajouter dans notre modeleur d'utiliser des calques afin de pouvoir superposer des cartes de terrain et des plans de ville existant par exemple. Il sera alors possible de modéliser plus fidèlement et simplement des environnements réels, bien que cet objectif n'était pas initialement le nôtre. Par ailleurs, les calques s'avèrent utiles lorsqu'il faut modéliser plusieurs niveaux du réseau urbain. Ces différents niveaux concernent notamment des réseaux différenciés

[1] http ://www.allegorithmic.com/
[2] http ://www.profxengine.com/

tels que le métro et les voies ferrées, mais aussi les différents niveaux d'un même réseau. En effet, il est courant de voir une autoroute passer au dessus d'une route moins importante sans créer pour autant d'intersection. L'ajout de calques nous permettra ainsi de modéliser ces phénomènes, ainsi que les jonctions autoroutières complexes.

Décomposition

Nous avons proposé de séparer le problème de la modélisation urbaine en deux problèmes distincts : celui de la décomposition et celui de l'évolution. En ce qui concerne la décomposition, une première étude, présentée dans cette thèse, s'attache à décrire la subdivision et la décomposition des îlots urbains en parcelles. Par ailleurs, le problème du placement de bâtiments sur ces parcelles a été abordé. Il reste encore beaucoup de travail sur ces problèmes précis. D'une part puisqu'il faut définir les processus socio-culturels influant sur de telles décompositions, et déterminer leurs paramètres respectifs. D'autre part, de nombreuses pistes sont ouvertes sur la diversité des décompositions et placements sur les parcelles, de la détermination de la nature de l'occupation des sols (usage commercial, résidentiel, etc.), etc.

Évolution

En ce qui concerne l'évolution, de nombreuses perspectives sont ouvertes. Une perspective particulièrement intéressante concerne les techniques de construction et leur évolution. En effet, nous avons montré que certaines techniques de construction peuvent être modélisées par un ensemble de règles de réécriture. Néanmoins, nous n'avons pas étudié la mutation de ces systèmes de réécriture. En effet, de nombreuses études proposent des méthodes pour simuler la mutation de génotypes représentés par des grammaires. Il serait intéressant d'appliquer ces travaux à des systèmes de réécriture simulant la construction d'un mur par exemple. On pourrait alors imaginer la combinaison d'une méthode de construction de murs en brique, ordonnée et régulière, et d'une méthode de construction d'un mur de pierres irrégulières. Cette étude nous permettrait en effet de mieux comprendre l'apparition et l'évolution des techniques de construction. Par ailleurs, il est intéressant de constater la corrélation entre les techniques de construction et les matériaux disponibles à l'endroit de leur découverte. Ceci suggère ainsi que ces règles sont géographiquement localisées.

Ceci nous amène à une autre de nos hypothèses de travail qui concerne justement la forme des réseaux urbains et les motifs qu'ils révèlent. En effet, de nombreux chercheurs, aujourd'hui, tiennent comme acquis l'existence *a priori* de tels motifs dans l'esprit des urbanistes et décideurs. Nous pensons, au contraire, qu'il s'agit, pour certains d'entre eux, de phénomènes émergents, tel l'orientation de la limaille de fer sur laquelle est appliquée un champ magnétique. Dans ce cas, les copeaux de fer s'orientent différemment en fonction de la forme des champs magnétiques présents. Dans le cadre des environnements urbain, la diversité des motifs en fonction des cultures peut s'expliquer par les différences dans les matériaux de construction, mais aussi par la nature et la forme des forces à l'œuvre. Ces forces peuvent être modélisées en utilisant des principes similaires aux champs magnétiques, en affectant à chaque élément de l'environnement urbain une certaine charge et ainsi un certain effet sur les autres éléments construits et à construire. La modification des charges pour le même objet dans des cultures différentes produit ainsi des motifs différents.

Par exemple, lorsque l'on réalise l'importance capitale des routes et de la voiture individuelle en Amérique du nord, est-il si surprenant de voir apparaître autant de lignes parallèles et de grilles ? De même, devant l'importance des organes de pouvoir (religieux et autres) dans l'Europe médiévale, n'est-il pas naturel de trouver autant de plan concentriques ? De tels modèles sont en cours d'expérimentation et promettent, tout au moins, des résultats intéressants.

Bibliographie

[AAAG95] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12) :752–761, 1995. 122

[AC98] M. Agarwal and Jonathan Cagan. A blend of different tastes : the language of coffee makers. *Environment and Planning B : Planning and Design*, 25(2) :205–226, 1998. 68

[Ad81] Harold Abelson and Andrea A. diSessa. *Turtle Geometry*. The MIT Press, Cambridge, MA, 1981. 57

[AIS75] C. Alexander, S. Ishikawa, and M. Silverstein. *The Oregon Experiment*, volume 3 of *Center for Environmental Structure Series*. Oxford University Press, New York, NY, 1975. 16

[AIS77] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language*, volume 2 of *Center for Environmental Structure Series*. Oxford University Press, New York, NY, 1977. 16

[Ale79] C. Alexander. *The Timeless Way of Building*, volume 1 of *Center for Environmental Structure Series*. Oxford University Press, New York, NY, 1979. 16

[All04] Rémy Allain. *Morphologie urbaine*. Arman Colin, Paris, 2004. 12, 17, 18, 115

[Aur91] Franz Aurenhammer. Voronoi diagrams
u2014a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3) :345–405, 1991. 121

[Aut01] Jacques Autran. Modélisation de l'information spatio-historique pour l'analyse et le projet en milieu patrimonial. Technical report, MAP-GAMSAU/CNRS, 2001. 38, 49

[BA05] Daniel Bekins and Daniel G. Aliaga. Build-by-number : Rearranging the real world to visualize novel architectural spaces. In *IEEE Visualization*, page 19, 2005. 70, 71, 156

[Bar53] Roland Barthes. *Le degré zero de l'écriture*. Seuil, Paris, 1953. 12, 40

[Bar64] Roland Barthes. *Éléments de sémiologie*. Denoël/Gonthier, Paris, 1964. 40

[Bar80] Roland Barthes. *La Chambre claire*. Gallimard/Seuil/Cahiers du cinéma, 1980. 40

[Ben99] Peter J. Bentley, editor. *Evolutionary Design by Computers*. Morgan Kaufmann, may 1999. 74, 76

[BFH05] Rene Berndt, Dieter W. Fellner, and Sven Havemann. Generative 3d models : a key to more information within less bandwidth at higher quality. In Nigel W. John, Sandy Ressler, Luca Chittaro, and David A. Duce, editors, *Web3D*, pages 111–121. ACM, 2005. 72

[BGT75] Philippe Boudon, Jacques Guillerme, and René Tabouret. *Figuration graphique en architecture*. D.G.R.S.T, A.R.E.A., Paris, 1975. 40, 45, 46

[BJPW99] A.C. Bailey, A.H. Jamson, A.M. Parkes, and S. Wright. Recent and future development of the leeds driving simulator. In *DSC'99*, Paris, July 1999. Driving Simulation Conference. 32, 49

[Bla95] J-Y. Blaise. L'objet architectural du point de vue de la mesure. *Rencontre des doctorants des écoles d'architecture du sud de la France*, juin 1995. 35, 37, 49, 155

[Bou95] B. Bouchareb. Le projet remus. développement d'un générateur de toitures. *Rencontre des doctorants des écoles d'architecture du sud de la France*, juin 1995. 38, 49

[Bou03] Philippe Boudon. *Sur l'espace architectural*. Éditions Parenthèses, second edition, 2003. 13, 14, 15, 45

[BPP02] Anne Bretagnolle, Fabien Paulus, and Denise Pumain. Time and space scales for measuring urban growth. *Cybergeo*, page 12, 2002. 18

[BT01] Michael Batty and Paul M. Torrens. Working paper 36 : Modeling complexity : The limits to prediction. Technical report, Centre for Advanced Spatial Analysis, University College London, London, UK, october 2001. 67

[Cag01] Jonathan Cagan. Engineering shape grammars : where we have been and where we are going. *Formal engineering design synthesis*, pages 65–92, 2001. 68

[CC78] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6) :350–355, November 1978. 72

[CCL04] Bertrand Coüasnon, Jean Camillerapp, and Ivan Leplumey. Making handwritten archives documents accessible to public with a generic system of document image analysis. In *DIAL*, pages 270–277. IEEE Computer Society, 2004. 102

[Cho63] Noam Chomsky. Formal properties of grammars. In Luce, Bush, and Galanter, editors, *Handbook of Mathematical Psychology 2*, pages 323–418. Wiley and Sons, New York, 1963. 53

[Com04] Jean Combaz. *Utilisation de phenomenes de croissance pour la generation de formes en synthese d'images*. PhD thesis, UJF, may 2004. 67

[Cor23] Le Corbusier. *Vers une architecture*. Flammarion, Paris, 1923. 69

[Cor24] Le Corbusier. *Urbanisme*. Éditions Vincent, Fréal and C., Paris, 1924. 17

[Cor30] Le Corbusier. *Précisions sur un état présent de l'architecture et de l'urbanisme*. Crès, Paris, 1930. 17, 114

[DAHF03] A. M. Day, David B. Arnold, Sven Havemann, and Dieter W. Fellner. Combining polygonal and subdivision surface approaches to modelling of urban environments. In *CW*, pages 189–197, 2003. 72

[DAHF04] A. M. Day, David B. Arnold, Sven Havemann, and Dieter W. Fellner. Combining polygonal and subdivision surface approaches to modelling and rendering of urban environments. *Computers and Graphics*, 28(4) :497–507, 2004. 72, 73, 76, 156

[Dal03] Ruth Conroy Dalton. The secret is to follow your nose : Route path selection and angularity. *Environment and Behavior*, 35 :107–131, 2003. 17

[dAMC+02] Cláudia Maria de Almeida, Antonio Miguel Vieira Manteiro, Gilberto Câmara, Britaldo Silveira Soares-Filho, Gustavo Countinho Cerqueira, Cássio Lopes Pennachin, and Michael Batty. Working paper 42 : Empiricism and stochastics in cellular automaton modeling of urban land use dynamics. Technical report, Centre for Advanced Spatial Analysis, University College London, London, UK, february 2002. 67

[DB03] Ruth Conroy Dalton and Sonit Bafna. The syntactical image of the city : A reciprocal definition of spatial elements and spatial syntaxes. In *4th International Space Syntax Symposium*, London, 2003. 17

[DFT04] Marie-Laurence Dekeersmaecker, Pierre Frankhauser, and Isabelle Thomas. Fractal dimension versus density of the built-up surfaces in the periphery of brussels. ERSA conference papers ersa04p69, European Regional Science Association, August 2004. available at http ://ideas.repec.org/p/wiw/wiwrsa/ersa04p69.html. 18

[DGdL+03] Fabien Dekeyser, François Gaspard, Livio de Luca, Michel Florenzano, Xin Chen, and Pascal Leray. Relevé du patrimoine architectural par relevé laser, vision par ordinateur, et exploitation des règles architecturales. In *Maquette Virtuelle et Patrimoine*, Cluny, march 2003. 36, 37, 49, 155

[DHL+98] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. *Computer Graphics*, 32(Annual Conference Series) :275–286, 1998. 53, 58, 103, 110

[Dij72] Edsger W. Dijkstra. The humble programmer. *Commun. ACM*, 15(10) :859–866, 1972. 73, 138

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B : Formal Models and Sematics (B)*, pages 243–320. North-Holland, Amsterdam, 1990. 105

[dM00] Jean-Marie Pérouse de Montclos. *Architecture, méthode et vocabulaire*. éditions du Patrimoine, Ministère de la Culture et de la Communication, Paris, 3rd edition, 2000. 15

[DOCJ91] A. Dagorne, J-Y. Ottavi, J-M. Castex, and M. Julian. Systèmes d'informations géographiques et gestion du territoire. In *actes du colloque SIG-GIS CARTAO*, 1991. 28, 49

[Don92] Stéphane Donikian. *Une approche déclarative pour la création de scènes tridimentionnelles : application à la conception architecturale*. PhD thesis, université de Rennes 1, 1992. 38, 49

[Don97] Stéphane Donikian. Vuems : a virtual urban environment modeling system. In *Computer Graphics International*, pages 84 – 92. CGI'97, June 1997. 33, 49, 139

[Don02] Stéphane Donikian. Les systèmes d'information géographique et la 3D pour la représentation de données urbaines. Technical report, Irisa/CNRS, 2002. 31

[Don04] Stéphane Donikian. *Modélisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environnements informés et structurés*. PhD thesis, Universit'e de Rennes 1, IFSIC (Habilitation), 2004. 33, 34, 45, 49, 139, 155

[DTC00] A. Dick, P. Torr, and R. Cipolla. Automatic 3d modeling of architecture. In *BMVC2000*, volume 1, pages 372–381, 2000. 23, 49

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs : A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series) :11–20, 1996. 26, 27, 49, 70, 155

[Duc05] Estelle Ducom. *La théorie des ceintures limitrophes (fringe belts) : une application aux villes françaises*. PhD thesis, Université de Rennes 2 Haute Bretagne, 2005. 17

[eC89] Division Informatique et Cartographie. Mise en œuvre de systèmes d'information urbains. Technical report, Services Techniques de l'Urbanisme, 1989. 28

[Eco72] Umberto Eco. *La structure absente, introduction à la recherche sémiotique*. Mercure de France, édition originale 1968, pour la traduction française 1972. 17, 40, 45

[Eco92] Umberto Eco. *Le signe, histoire et analyse d'un concept*. Le livre de poche, Paris, mai 1992. 40

[Ein21] Albert Einstein. *La géométrie et l'expérience*. Gauthier-Villars et cie, 1921. Traduction de Geometrie und Erfahrung par Maurice Solovine. 14

[EK96] M. Engeli and D. Kurmann. A virtual reality design environment with intelligent objects and autonomous agents. In *Design and Decision Support Systems*, Spa Belgium, 1996. 45

[EMP+03] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling : A Procedural Approach, 3rd edition, with contributions from W.R. Mark and J.C. Hart*. Morgan Kaufmann, Elsevier Science, USA, january 2003. 52, 72, 103

[EWU97] Guy Engelen, Roger White, and Inge Uljee. *Decision Support Systems in Urban Planning*, chapter Integrating Constrained Cellular Automata Models, GIS and Decision Support Tools for Urban Planning and Policy Making. Taylor and Francis Group, december 1997. 67

[Fau01] D. Faucher. Urbanlab, modélisation déclarative des enveloppes urbaines réglementaires. thèse de l'ecole doctorale des sciences pour l'ingénieur de nantes, 2001. 38, 49

[Fay99] Philippe Fayeton. Écrire dans la ville, prolégomènes à une pédagogie de l'architecture urbaine. résumé de thèse en cours à l'ufr lacs langage et parole de l'université d'aix-marseille 1, 1999. 17, 40

[Fay01] Philippe Fayeton. Écrire dans la ville, prolégomènes à une pédagogie de l'architecture urbaine. thèse de sciences de l'information et de la communication de l'université de lyon 2, 2001. 40

[Fay02] Philippe Fayeton. Ecrire la ville, 2002. 17, 40

[FBD97] M. Florenzano, J-Y. Blaise, and P. Drap. Paros, modèles objet appliqués à l'étude de l'architecture construite. *L'Objet*, 3(1) :29–52, Avril 1997. 35, 49

[FBD00] Michel Florenzano, Jean-Yves Blaise, and Iwona Dudek. Le programme arkiw. un système d'information et de représentation des connaissances relatives aux édifices patrimoniaux et à leurs évolutions architecturales. le cas du rynek gówny à cracovie. Technical report, MAP Gamsau, Institut Historii Architektury i Konserwacji Zabytków Wydzial Architektury Politechniki Krakowskiej (HAiKZ WA PK), 2000. 35, 49

[FEC02] R. Fabbri, L. F. Estrozi, and L. Da F. Costa. On voronoi diagrams and medial axes. *J. Math. Imaging Vis.*, 17(1) :27–40, 2002. 121, 122

[Fer99] R. S. Ferrero. Ls sketchbook, a graphical environment for the creation of l-systems. *University of Oviedo, Spain*, 1999. 58

[Fij02] Yankel Fijalkow. *Sociologie de la ville*. Repères. La découverte, Paris, 2002. 18

[Fle87] U. Flemming. More than the sum of the parts : the grammar of queen anne houses. *Environment and Planning B : Planning and Design*, 14 :323–350, 1987. 67

[FML06] D. Fradin, D. Meneveaux, and P. Lienhardt. A hierarchical topology-based model for handling complex indoor scenes. *Computer Graphics Forum*, 25(2) :149–162, jun 2006. 138

[Fue97] K. Fuesser. City, roads and traffic. *Vieweg Verlag*, 1997. 62

[FZ03] Christian Früh and Avideh Zakhor. Constructing 3d city models by merging ground-based and airborne views. In *CVPR (2)*, pages 562–569, 2003. 23, 49

[Gib74] James J. Gibson. *The perception of the visual world*. Greenwood Press, 1974. 46

[Gip74] J. Gips. Shape grammars and their uses. phd thesis, stanford university, 1974. 67

[Gre89] N. Greene. Voxel space automata : modeling with stochastic growth processes in voxel space. In *SIGGRAPH '89 : Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 175–184, New York, NY, USA, 1989. ACM Press. 67

[Gru03] Alain Grumbach. *Cognition Virtuelle : Réflexion sur le virtuel, ses implications cognitives, ses réalisations artistiques*. ouvrage électronique disponible sur : http ://www.enst.fr/ grumbach/cognition-virtuelle, GET/ENST Paris, 2 edition, 2003. 45

[Hab00] N. John Habraken. *The Structure of the Ordinary : Form and Control in the Built Environment*. The MIT Press, 2000. 17, 115, 116, 121, 157

[Hal71] Edward T. Hall. *La dimension cachée*. Le Seuil, édition originale 1966, pour la traduction fraçaise 1971. 18

[Har92] John C. Hart. The object instancing paradigm for linear fractal modeling. In *Proceedings of the conference on Graphics interface '92*, pages 224–231. Morgan Kaufmann Publishers Inc., 1992. 53, 103, 155

[Har94] John C. Hart. On efficiently representing procedural geometry, 1994. 53, 155

[HB99] N. Haala and C. Brenner. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 130–137, 1999. 25, 49

[HBS+99] S. Hinz, A. Baumgartner, C. Steger, H. Mayer, W. Eckstein, H. Ebner, and B. Radig. Road extraction in rural and urban areas. In Wolfgang Förstner, Claus-Eberhard Liedtke, and Jürgen Bückner, editors, *Semantic Modeling for the Acquisition of Topographic Information from Images and Maps*, pages 133–153, 1999. 25, 49

[HF01] Sven Havemann and Dieter Fellner. A versatile 3d model representation for cultural reconstruction. In *VAST '01 : Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*, pages 205–212, New York, NY, USA, 2001. ACM Press. 72, 156

[HF03] Sven Havemann and Dieter W. Fellner. Generative mesh modeling. Technical Report TUBSCG-2003-01, Institute of ComputerGraphics, TU Braunschweig, Braunschweig, Germany, 2003. 72

[HF04] Sven Havemann and Dieter W. Fellner. Generative parametric design of gothic window tracery. In *SMI*, pages 350–353, 2004. 72, 76

[Hil96] Bill Hillier. *Space is the Machine*. Cambridge University Press, Cambridge, 1996. 17

[HJA02] Christoph M. Hoffmann and R. Joan-Arinyo. *Handbook of Computer Aided Geometric Design*, chapter Parametric Modeling. Elsevier, North-Holland, 2002. 46

[HYN03] Jinhui Hu, Suya You, and Ulrich Neumann. Approaches to large-scale urban modeling. *IEEE Computer Graphics and Applications*, 23(6) :62–69, 2003. 26, 49

[Inc99] Adobe Systems Inc. *PostScript Language Reference*. Addison-Wesley, 3 edition, 1999. 72

[Jac61] Jane Jacobs. *The death and life of great American cities*. Vintage Books USA, 1961. 16

[Jac02] Helen Jackson. *Creative evolutionary systems*, chapter 11, Toward a symbiotic coevolutionary approach to architecture, pages 299–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. 74

[JDPM] H. Jibrini, M. Pierrot Deseilligny, N. Paparoditis, and H. Maître. Reconstruction 3-d de bâtiments à partir de données 2-d cadastrales vectorisées et d'images aériennes. 25, 49

[JLF96] William Jepson, R. Liggett, and Scott Friedman. Virtual modeling of urban environments. *Presence : Teleoperators and Virtual Environments*, 5(1), 1996. MIT Press. 31, 49, 155

[JM99] W. Jepson and R. Muntz. A real-time visualisation system for large scale urban environments. In *actes de VwSim'99*, San Francisco, janvier 1999. SCS Western Multi-Conference. 31, 49

[Kau84] Stuart A. Kauffman. Emergent properties in random complex automata. *Physica D : Nonlinear Phenomena*, 1984. 67

[KE96] D. Kurmann and M. Engeli. Modeling virtual space in architecture. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pages 77–82, 1996. 45

[Kli96] Jean-Marie Klinkenberg. *Précis de sémiotique générale*. "Points Essais" série "Sciences humaines". De Boeck Université, 1996. 39

[Kni99] Terry Knight. Applications in architectural design, and education and practice. In *NSF/MIT Workshop on Shape Computation*, April 1999. 67

[Kni00] Terry W. Knight. Shape grammars in education and practice : History and prospects. *International Journal of Design Computing*, 2, 2000. 67, 76

[KP03] Radoslaw Karwowski and Przemyslaw Prusinkiewicz. Design and implementation of the L+C modeling language. In Jean-Louis Giavitto and Pierre-Etienne Moreau, editors, *Electronic Notes in Theoretical Computer Science*, volume 86. Elsevier, 2003. 102

[Kur94] Winfried Kurth. Growth grammar interpreter grogra 2.4 - a software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modelling. In *Bericbte des Forschungszentrums Waldökosysteme der Universität Göttingen*, 1994. 58

[Ley01a] Michael Leyton. A generative theory of shape. *Lecture Notes in Computer Science*, 0(2145) :0, 2001. 71, 156

[Ley01b] Michael Leyton. Group theory and architecture 1 : Nested symmetries. *Nexus Network Journal*, 3(3), summer 2001. 71

[Ley01c] Michael Leyton. Group theory and architecture 2 : Why symmetry/asymmetry ? *Nexus Network Journal*, 3(4), autumn 2001. 71

[LGJ05] Gurvan Le Guernic and Thomas Jensen. Monitoring information flow. In Andrei Sabelfeld, editor, *Proceedings of the 2005 Workshop on Foundations of Computer Security*. DePaul University, June 2005. LICS'05 Affiliated Workshop. 105

[LGP05] Gurvan Le Guernic and Julien Perret. Fl-system's intelligent cache. In Alexandre Vautier and Sylvie Saget, editors, *Proceedings of Majecstic 2005*, pages 79–88, Rennes, november 2005. 103, 107, 108, 110

[Liè97] S. Liège. Modélisation déclarative incrémentale. application à la conception urbaine. Technical report, Ecole des Mines de Nantes, 1997. 38, 49

[Lin68] A. Lindenmayer. Mathematical models for cellular interactions in development, I and II. *J. Theor. Biol.*, 18 :280–315, 1968. 53, 76

[LML+00] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, and A. Baumgartner. Automatic extraction of roads from aerial images based on scale-space and snakes. *Machine Vision and Applications*, 12 :23–31, 2000. 25, 49

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project : 3D scanning of large statues. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000. 24, 49

[Lyn60] K. Lynch. *The Image of the City*. MIT Press, Cambridge/MA, 1960. 16, 115

[Mah95] Mustapha Ben Mahbous. Un système d'information en architecture et urbanisme basé sur la représentation de connaissance. *Rencontre des doctorants des écoles d'architecture du sud de la France*, juin 1995. 28, 37, 49

[Mar01] J-E. Marvie. Visualisation temps-réel de scènes complexes dans un environnement distribué. rapport de dea informatique. irisa, université de rennes 1, 2001. 88

[Mar04] Jean-Eudes Marvie. *Visualisation Interactive d'Environnements Virtuels Complexes à travers des Réseaux et sur des Machines à Performances Variables*. PhD thesis, INSA de Rennes, Octobre 2004. 88, 96

[MBST01] Xavier Marsault, Christophe Bertrand, Renato Saleri, and Joëlle Thollot. Bases de données urbaines 3d complexes. modélisation et restitution. Technical report, MAP-Aria, équipe iMAGIS/GRAVIR, 2001. 32, 49, 65, 74, 76, 155

[MCB97] Chris Marrin, Rikk Carey, and Gavin Bell. A vrml specification. Technical report, VRML consortium, 1997. 53, 85

[Mit90] William J. Mitchell. *The Logic of Architecture, Design, Computation and Cognition*. MIT Press, Cambridge, Massachusetts. London, England, 1990. 38, 69

[MPB03] Jean-Eudes Marvie, Julien Perret, and Kadi Bouatouch. Remote interactive walkthrough of city models using procedural geometry. Technical report, Irisa, campus de Beaulieu, Rennes, France, 2003. 32p. 88, 89, 96

[MPB05] J.-E. Marvie, J. Perret, and K. Bouatouch. The fl-system : A functional l-system for procedural geometric modeling. *The Visual Computer*, 2005. 70, 88, 89, 96

[MWH+06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *SIGGRAPH '06 : ACM SIGGRAPH 2006 Papers*, pages 614–623, New York, NY, USA, 2006. ACM Press. 69, 70, 156

[MZ] Montiel and Zisserman. Automated architectural acquisition from a camera undergoing planar motion. 23, 49

[MZ94] J-L. Maltret and J. Zoller. Models of terrains for building placement. In *International Kolloquium fur anwendungen der Mathematik in architektur und bauwesen*, Weimar, Février 1994. 38, 49

[MZ96] J-L. Maltret and J. Zoller. Simulation of architectural and urban morphology. In *OEEPE Workshop on 3D-city models*, Bonn, Octobre 1996. 38, 49

[OR23] C. K. Ogden and I. A. Richards. *The Meaning of Meaning. A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Harcourt, Brace and Brace, New York, 1923. 39

[OT00] David O'Sullivan and Paul Torrens. Working paper 22 : Cellular models of urban systems. Technical report, Centre for Advanced Spatial Analysis, University College London, London, UK, june 2000. 67

[PABCV00] Jean-Pierre Perrin, Najla Allani-Bouhoula, Christine Chevrier, and Emmanuel Viard. Projet medina : Reconstruction de volumétries urbaines. Technical report, MAP, 2000. 38, 49, 155

[Pal97] Andrea Palladio. *Les quatre livres de l'Architecture*. Flammarion, Paris, deuxième edition, 1997. 15, 100

[PDD99] Philippe Panerai, Jean-Charles Depaule, and Marcelle Demorgon. *Analyse urbaine*. collection eupalinos. Éditions Parenthèses, Marseilles, 1999. 18

[Pen03] Alan Penn. Space syntax and spatial cognition : Or why the axial line ? *Environment and Behavior*, 35 :30–65, 2003. 17

[Per03] Frank Perbet. Dynamic graph : un outil générique pour la modélisation multi-échelle. In *Proceedings of the 16th Journée AFIG 2003*, Paris, 2003. 52, 73, 76

[PJM94] Przemyslaw Prusinkiewicz, Mark James, and Radomír Měch. Synthetic topiary. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 351–358. ACM Press, 1994. 69

[PKVG] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Automated reconstruction of 3d scenes from sequences of images. 26, 49

[PLH+90] P. Prusinkiewicz, A. Lindenmayer, J. S. Hanan, et al. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990. 54, 57, 58, 59, 60, 83, 155, 156

[PM01] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of SIGGRAPH'01*, pages 301–308, Los Angeles, CA, USA, August 2001. ACM. 60, 61, 62, 63, 64, 65, 66, 75, 76, 120, 156

[PMKL01a] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 289–300, New York, NY, USA, 2001. ACM Press. 57

[PMKL01b] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 289–300. ACM Press, 2001. 57

[Pop91] Karl Popper. *La connaissance objective*. Aubier, Paris, édition originale de 1972, pour la traduction française 1991. Traduction complète de J.-J. Rosat. 14

[Pru03] Przemyslaw Prusinkiewicz. L-systems and beyond. In *Course notes from SIGGRAPH*. ACM, 2003. 54, 55, 56, 57, 85, 99, 155

[Pum04] Denise Pumain. Scaling laws and urban systems. Working Paper no 04-02-002, 2004. Santa Fe Institute. 18

[Rec05] Alex Reche. *Image-Based Capture and Rendering with Applications to Urban Planning*. PhD thesis, Universite de Nice Sophia-Antipolis, April 2005. 27, 49

[Roc06] Claude Rochet. Sciences de la complexité et philosophie politique appliquées à la transformation des organisations, 2006. http ://perso.wanadoo.fr/claude.rochet/plan.html. 12

[Ron02] Marcel Roncayolo. *Lectures de villes, formes et temps*. Eupalinos. Parenthèses, 2002. 17

[Ros97] M. Rosenman. The generation of form using an evolutionary approach, 1997. 74

[Sal99] Nikos A. Salingaros. Architecture, patterns, and mathematics. *Nexus Network Journal*, 1(2), 1999. 17

[Sal00] Nikos A. Salingaros. Complexity and urban coherence. *Journal of Urban Design*, 5 :291–316, 2000. 17

[Sal03] Renato Saleri. Pseudo-urban automatic pattern generation. In *INSC 2003 , International Nonlinear Sciences Conference 2003*, New York, february 2003. University of Vienna, Austria, Chaos and Complexity Letters (CCL) Nova Science. Co-edited by the academy of architecture, University of italian switzerland, Mendrisio, Italy. 74, 75, 156

[SG72] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In C V Freiman, editor, *IFIP Congress71*, pages 1460–1465, Amsterdam : North-Holland, 1972. 67

[SG96] T. Schnier and J. Gero. Learning genetic representations as alternative to hand-coded shape grammars, 1996. 74

[Sin03] Gary Singh. Modeling cities one segment at a time. *IEEE Computer Graphics and Applications*, 23(6) :4–5, 2003. 24, 49

[SM78] G. Stiny and W. J. Mitchell. The palladian grammar. *Environment and Planning B : Planning and Design*, 5 :5–18, 1978. 67

[Smi84] Alvy Ray Smith. Plants, fractals, and formal languages. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 1–10. ACM Press, 1984. 52

[Sti75] George Stiny. *Pictorial and Formal Aspects of Shape and Shape Grammars*. Prof. Salomon Klaczko-Ryndziun, Frankfurt a. M., Birkhauser Verlag, Basel, 1975. 67

[Tap92] Mark Tapia. Chinese lattice designs and parametric shape grammars. *Vis. Comput.*, 9(1) :47–56, 1992. 68

[Tho99] Gwenola Thomas. *Environnements virtuels urbains : modélisation des informations nécessaires à la simulation d'humanoïdes*. PhD thesis, Université Rennes I, IRISA, Rennes, 1999. 33, 49, 139

[Tho05] Romain Thomas. *Modèle de mémoire et de carte cognitive spatiales : application à la navigation du piéton en environnement urbain*. PhD thesis, Université Rennes I, IRISA, Rennes, 2005. 17, 45

[Tor00] Paul M. Torrens. Working paper 28 : How cellular models of urban systems work (1. theory). Technical report, Centre for Advanced Spatial Analysis, University College London, London, UK, november 2000. 67, 75

[Tur91] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91 : Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1991. ACM Press. 66

[VP95] Vitruve and Claude Perrault. *Les dix livres d'architecture*. Architecture Urbanisme. Mardaga, 2ème edition, 1995. 15, 100

[vVDBB98] Hendrik A. H. C. van Veen, Hartwig K. Distler, Stephan J. Braum, and Heinrich H. Bülthoff. Navigating through a virtual city : using virtual reality technology to study human action and perception. *Future Gener. Comput. Syst.*, 14(3-4) :231–242, 1998. 30, 49, 155

[WK91] Andrew Witkin and Michael Kass. Reaction-diffusion textures. *Computer Graphics*, 25(4) :299–308, 1991. 66

[WWSR03] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Transaction on Graphics*, 22(3) :669–677, July 2003. 68, 69, 76, 78, 156

[Zev59] Bruno Zevi. *Apprendre à voir l'architecture*. Forces vives. Éditions de minuit, Paris, 1959. 13

[Zev91] Bruno Zevi. *Langage moderne de l'architecture*. Agora. POCKET, DUNOD, Paris, 1991. 15

[ZS01] H. Zhao and R. Shibasaki. Reconstructing urban 3d model using vehicle-borne laser range scanners. In *3DIM2001*, Québec, Canada, mai 2001. 23, 49

Table des figures

Résumé

Le problème de la modélisation des environnements urbains virtuels reste complexe. Les principaux obstacles à cette modélisation sont liés à la complexité et à la grande quantité de données que les environnements urbains représentent. En effet, le volume et la complexité de ces données pose d'une part des problèmes de stockage et de manipulation, et complexifie d'autre part leur visualisation et leur mise à jour. Dans cette thèse, nous proposons une meilleure utilisation des connaissances *a priori* disponibles sur les environnements urbains. Tout d'abord, nous présentons un système de réécriture original, les *FL-systems*, comme une nouvelle extension fonctionelle des *L-systems*, qui sont principalement utilisés pour la modélisation de processus biologiques. Les *FL-systems*, grâce à l'utilisation de fonctions comme terminaux du langage, permettent une génération en temps-réel d'objets géométriques. Par ailleurs, nous proposons un mécanisme de gestion de cache pour la réécriture des *FL-systems* afin de réutiliser de façon intelligente les modules déjà réécrits. Pour finir, nous définissons une nouvelle méthode de modélisation des environnements urbains à l'aide de hiérarchies d'abstraction, qui permettent de focaliser la modélisation sur chaque niveau de détail de la ville, et d'algorithmes de subdivision de polygones simples.

Abstract

Modeling virtual urban environments remains a complex problem. The main issues to model such environments are related to the great complexity and huge amount of data they represent. In fact, this amount of data is hard to store and manipulate, but also to visualize and update. In this thesis, we propose a better usage of *a priori* knowledge on urban environments. First, we present an original rewriting system, *FL-systems*, as a functional extension of *L-systems*, mainly used for biologically based modeling. Terminal elements being functions, *FL-systems* allow the real-time generation of geometric objects. Furthermore, we propose a new cache mechanism for *FL-systems* in order to reuse previously rewritten modules. Finally, we define a new modeling method for urban environments using abstraction hierarchies and simple polygon subdivision algorithms, which allows to focus the modeling attention on each level of detail of the city separately.